2

NCS TIB 90-12

NATIONAL COMMUNICATIONS SYSTEM

TECHNICAL INFORMATION BULLETIN 90-12

INVESTIGATION OF THE PROGRESSIVE BI-LEVEL CODING TECHNIQUE FOR THE HIGH-RESOLUTION BI-LEVEL DATA COMPRESSION STANDARD

JULY, 1990

OFFICE OF THE MANAGER
NATIONAL COMMUNICATIONS SYSTEM

WASHINGTON, D.C. 20305

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
July 1990

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Investigation of the Progressive Bi-Level Coding Technique for the High-Resolution Bi-Level Data Compression Standard

5. FUNDING NUMBERS

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Delta Information Systems, Inc.
Horsham Business Center, Bldg. 3
300 Welsh Road, Horsham, PA 19044

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Communications System
Office of Technology & Standards
Washington, DC 20305-2010

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NCS TIB 90-12

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this study was to investigate the Progressive Bi-Level Coding Technique for the High Resolution Bi-Level Data Compression Standard. At the present time, CCITT Recommendations for Group 4 permit the transmission of only sequentially transmitted black and white imagery. In addition, coding efficiency of Group 4 machines suffers for input pages containing half-time information. As a result of much increased commercial interest from major computer and telecommunications companies, there has been intense effort in the international standards bodies to select a bi-level image compression technique for future image storage and communications applications. The focal point of this activity has been the Joint Bi-Level Image Group (JBIC) of ISO/IEC and CCITT.

14. SUBJECT TERMS
Group 4
Facsimile
Progressive Bi-Level Coding Technique
Data Compression

15. NUMBER OF PAGES
107

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet *optical scanning requirements.*

Block 1. Agency Use Only *(Leave blank)*

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	-	Contract	PR	-	Project
G	-	Grant	TA	-	Task
PE	-	Program Element	WU	-	Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as. Prepared in cooperation with..., Trans. of..., To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities.
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.
DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA - Leave blank.
NTIS - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only).*

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

NCS TECHNICAL INFORMATION BULLETIN 90-12

INVESTIGATION OF THE PROGRESSIVE BI-LEVEL CODING TECHNIQUE FOR THE HIGH-RESOLUTION BI-LEVEL DATA COMPRESSION STANDARD

JULY 1990

PROJECT OFFICER

Stephen Perschau
STEPHEN PERSCHAU
Computer Scientist
Office of NCS Technology
and Standards

APPROVED FOR PUBLICATION:

Dennis Bodson
DENNIS BODSON
Assistant Manager
Office of NCS Technology
and Standards

FOREWORD

Among the responsibilities assigned to the Office of the Manager, National Communications System, is the management of the Federal Telecommunication Standards Program. Under this program, the NCS, with the assistance of the Federal Telecommunication Standards Committee identifies, develops, and coordinates proposed Federal Standards which either contribute to the interoperability of functionally similar Federal telecommunication systems or to the achievement of a compatible and efficient interface between computer and telecommunication systems. In developing and coordinating these standards, a considerable amount of effort is expended in initiating and pursuing joint standards development efforts with appropriate technical committees of the Electronics Industries Association, the American National Standards Institute, the International Organization for Standardization, and the International Telegraph and Telephone Consultative Committee of the International Telecommunication Union. This Technical Information Bulletin presents an overview of an effort which is contributing to the development of compatible Federal, national, and international standards in the area of Video Teleconferencing. It has been prepared to inform interested Federal activities of the progress of these efforts. Any comments, inputs or statements of requirements which could assist in the advancement of this work are welcome and should be addressed to:

Office of the Manager
National Communications System
ATTN: NCS-TS
Washington, DC 20305-2010

INVESTIGATION OF THE PROGRESSIVE BI-LEVEL CODING TECHNIQUE FOR THE HIGH-RESOLUTION BI-LEVEL DATA COMPRESSION STANDARD

JULY, 1990

SUBMITTED TO:
NATIONAL COMMUNICATIONS AGENCY
Office of Technology and Standards
WASHINGTON, D.C. 20305

Contracting Agency:
DEFENSE COMMUNICATIONS AGENCY
Contract Number - DCA100-87-C-0078
Task Order Number - 89-C-0078-002

DELTA INFORMATION SYSTEMS, INC.
Horsham Business Center, Bldg. 3
300 Welsh Road
Horsham, PA 19044

TEL: (215) 657-5270

FAX: (215) 657-5273

TABLE OF CONTENTS

1.0 INTRODUCTION

This document summarizes work performed by Delta Information Systems, Inc., (DIS) for the National Communications System (NCS), Office of Technology and Standards. This office is responsible for the management of the Federal Telecommunications Standards Program, which develops telecommunications standards, whose use is mandatory for all Federal agencies. The purpose of this study, performed under task order number 89-C-0078-001 of contract number DCA100-89-C-0078, was to investigate the Progressive Bi-level Coding Technique for the High Resolution Bi-level Data Compression Standard.

The progressive bi-level coding technique consists of repeatedly reducing the resolution of a bi-level image, R_0, creating images R_1, R_2,..., R_n, image R_i having one-half the number of pixels per line and one-half the number of lines of image R_{i-1}. The lowest-resolution image, R_n, called the base layer, is transmitted losslessly (free of distortion) by binary arithmetic coding. Next, image R_{n-1} is transmitted losslessly, using pixels in R_n and previously-transmitted (causal) pixels in R_{n-1} as predictors in an attempt to predict the next R_{n-1} pixel to be transmitted. If prediction is possible (both transmitter and receiver are equipped with rules to tell whether this is the case), the predicted pixel value is not transmitted. This progressive build-up is repeated until image R_0 has been losslessly transmitted.

1.1 Background

At the present time, CCITT Recommendations for Group 4 permit the transmission of only sequentially transmitted black and white imagery. In addition, coding efficiency of Group 4 machines suffers for input pages containing half-tone information. As a result of much increased commercial interest from major computer and telecommunications companies, there has been intense effort in the international standards bodies to select a bi-level image compression technique for future image storage and communications applications. The focal point of this activity has been the Joint Bi-level Image Group (JBIG) of ISO/IEC and CCITT.

The JBIG was formed in 1988 under the umbrella of the ISO working group (now ISO/IEC/JTC1/SC2/WG8 - Coded Representation of Picture and Audio Information). It brings together ISO picture coding knowledge with CCITT telecommunication service expertise (from the New Image Communications (NIC) group of CCITT Study Group VIII). Its aim was to select and develop a compression/decompression technique for a general class of bi-level images. The technique will form the basis of both an ISO standard and a CCITT recommendation.

A specification for a compression technique was formulated for a potential range of services and applications including facsimile, audiographic teleconferencing and image databases. To support such a range of applications the technique should be adaptable to a wide range of image resolutions and to varying image quality. It should also be capable of providing progressive (multi-stage with improving quality) or sequential image build-up.

At a series of JBIG meetings, beginning in Stockholm in July, 1989, an algorithm was selected from the five remaining prior to the Stockholm meeting. The five were: Progressive Transmission of Binary Images by Hierarchical Coding (BIHC); Progressive Encoding of Facsimile Image Using Edge Decomposition (PED); Progressive Encoding of Predicted Signal According to Classified Pel Patterns (PCLAP); Progressive Adaptive Bi-level Image Compression (PBIC); and Progressive Coding Scheme Using Block Reduction (PCSB).

The selected algorithm performs image reduction, typical prediction (TP), deterministic prediction (DP) and binary arithmetic encoding and decoding. The image reduction algorithm, called PRES (Progressive Reduction Standard for Bi-level Images), is a combination of BIHC and PCSB with better reduced image quality than either.

The Progressive Bi-level Coding Technique is a coding technique showing the promise of large compression ratios for the coding and progressive transmission of bi-level data. No comprehensive study analyzing this technique, as applied to Group 4 facsimile systems under carefully controlled conditions, has been performed to date. The purpose of the task reported herein was to analyze the Progressive Bi-level Coding Technique to determine its relative effectiveness as applied to Group 4 facsimile.

This report is composed of six sections and two appendices. Section 1.0 presents a brief synopsis of the study and outlines its results and conclusions. Section 2.0 covers the theory and description of the JBIG Base

System. Section 3.0 describes the simulation system developed during this study. Section 4.0 presents the simulation objectives and briefly describes the test images. Section 5.0 covers the simulation results, and Section 6.0 presents the conclusions derived from these results. Appendix A outlines the basic principles of binary arithmetic coding, and Appendix B contains displays of the test images.

1.2 Synopsis

The investigation was conducted in four major, and to some extent overlapping, phases. The first phase consisted of a study of JBIG documentation, and continued throughout the project as more was released. Simulation software, hereafter called the study system, was developed in the second phase with the objectives of functionally duplicating the JBIG Base System and of supplementing data emerging from the ongoing JBIG investigations. The third phase was devoted to simulating the transmission and reception of JBIG test images. The fourth phase consisted of evaluating the results, writing the final report and preparing all deliverable items.

Two significant results emerged from this study: (1) best data compression usually was achieved by performing at least one image reduction rather than simply coding the highest-resolution image; and (2) all the major features of the algorithm played important roles in data compression, some more than others, depending upon image type and resolution.

2.0 INVESTIGATION

2.1 Theory

2.1.1 General Description

The Progressive Bi-Level Coding Technique consists of starting with a bi-level image of some resolution and repeatedly reducing it, each new image having half as many pixels per line and half as many lines as its parent image. (If the line length and/or number of lines is odd in the parent image, the corresponding number in the reduced image is rounded up to the nearest integer.) The lowest-resolution image is called the base layer; all others are called difference layers.

The base layer is transmitted without distortion, pixel by pixel, data compression achieved by binary arithmetic coding. The next higher resolution layer is then transmitted by a combination of prediction and binary arithmetic coding. Prediction is based on the values of predictor pixels from the base layer, all of whose pixels are known to both transmitter and receiver, and from difference layer pixels already encoded or predicted. Pixels which the transmitter and receiver both "know" can be predicted correctly are not encoded. Successively higher resolution images are similarly transmitted with the previous image used as a reference until the original image has been transmitted (or the process stopped at the receiver's request).

A sequential mode of transmission also exists. It consists of performing the entire progressive transmission on successive horizontal stripes of the original image. In this mode the receiving party never sees the entire image until all stripes have been completed.

2.1.2 Notations and Definitions

The following terms to be used throughout this report are defined here.

Layer — Any one image in the progressive hierarchy.

Base Layer — The lowest resolution image in the hierarchy.

Difference Layer — Image being encoded or decoded other than the base layer.

Reference Layer — The layer whose dimensions are half those of the difference layer currently being encoded or decoded. The has no reference layer.[1]

Template — A set of pixels in the reference and/or encoded (decoded) layer in the neighborhood of the pixel being processed. Templates are used for prediction and for binary arithmetic coding "contexts."

State — A binary number representing the values of the pixels in a template; also the current point in the probability estimation chain of the binary arithmetic coder.

AT — Adaptive context templates, described below.

DP — Deterministic prediction, described below.

TP — Typical prediction, described below.

2.1.3 Image Reduction

In all the image reduction algorithms proposed in various JBIG documents, including PRES, each low-resolution pixel is determined by the values of several high-resolution pixels and low-resolution pixels that have already been determined. The objective of the reduction algorithm is to preserve as much detail as possible in the low-resolution image under the constraint that the latter be half as wide and high as the high-resolution image. Different algorithms approach this objective in different ways; the PRES approach is described later.

2.1.4 Prediction

When a difference layer is being encoded or decoded, much of the compression is achieved by predicting new pixel values from the values of pixels in a predictor template. The predictor template contains pixels from the reference layer and pixels already predicted or encoded from the difference

[1] In image reduction, the higher-resolution layer is called the reference layer, since the lower-resolution layer is the layer being produced.

layer. When the predictor state is such that the prediction is known to be correct (the receiver must know this also), the predicted pixel value need not be encoded or decoded. The JBIG algorithm employs two kinds of prediction: typical prediction (TP) and deterministic prediction (DP). The meanings of TP and DP are described in general terms here, and their implementations in the JBIG algorithm are described in a later section.

2.1.4.1 Typical Prediction (TP)

Typical prediction refers to prediction in which the predicted value is almost always, but not necessarily always, correct. Since, in bi-level imagery, each pixel carries only one bit of information, it would be wasteful for the transmitter to inform the receiver of whether the prediction is correct for each pixel predicted. Instead, the transmitter looks ahead for and reports TP errors (exceptions). One reporting method is to transmit a pointer to the next exception. Another is to transmit an exception/no-exception bit at known intervals, the bit set if the interval following it contains at least one exception. If so, TP is disabled throughout that entire interval.

2.1.4.2 Deterministic Prediction (DP)

Deterministic prediction refers to prediction in which the predicted value is always correct. DP is tightly bound to the image reduction rules. Whether a pixel is or is not deterministically predictable is determined by looking up a rule in a table indexed by the state of the predictor pixels. The rule is one of the following: Predict black, Predict white or Don't predict.

2.1.5 Binary Arithmetic Coding

2.1.5.1 Binary Arithmetic Coders

Appendix A presents the basic principles of binary arithmetic coders. In a later section the JBIG coder is briefly described.

2.1.5.2 Contexts

In the description of binary arithmetic coding presented in Appendix A, it is assumed that the context (in the ordinary English sense) in which the symbol occurs is ignored.

The data compression achieved by a binary arithmetic coder is best when the probabilities of the two symbols are near 1 and 0, and worst when they are near 1/2. In any practical application, the probability of a 1 or 0 at any given time is frequently dependent upon the conditions under which the symbol is being encoded or decoded. Therefore, best compression is achieved by keeping separate probability estimates for those conditions under which the encoded symbol probabilities are the most strongly skewed. These conditions are called contexts.

Consider, for example, a bi-level image containing line drawings and text. As this image is scanned, if the previous pixel was white, then there is a high probability that the current one will be white also. Therefore, if one uses the previous pixel value as a predictor, there are two contexts, one for each color of the previous pixel. The probabilities for each are usually much nearer 1 and 0 than is the single probability with the previous pixel value ignored.

In the JBIG system, there is a separate context for every possible combination of pixel values in a context "template." These templates are described later. Because there can be thousands of contexts, an important consideration in system design is to minimize the memory requirements for each.

2.1.5.3 Adaptive Context Templates

The purpose of adaptive context templates is to take advantage of horizontal periodicity, which often occurs in half-tone images.

Data compression is best if at least one of the pixels in a context template is a good predictor of the pixel being encoded. An adaptive context template, bearing the acronym AT (adaptive template, formerly AC for adaptive context), contains a "floating" pixel; all other pixels in the template are fixed in position relative to the encoded pixel. There are also other pixels designated as candidate floating pixels, not currently a part of the context template.

If one of the candidates becomes a much better predictor than the current floating pixel, then the candidate and current floating pixels swap status, so that the better predictor becomes a part of the context template. This test is made infrequently, and the swap is made only if the candidate is a much better predictor. These restrictions are imposed because, when a swap is made, compression is temporarily degraded until the binary arithmetic coder has time to adapt to the swap.

2.2 Transmission Modes

The JBIG specification calls for two transmission modes: progressive and sequential.

2.2.1 Progressive Transmission

In progressive transmission the entire starting image is reduced to half its height and width. The reduced image is similarly reduced, this process repeated some specified number of times. The image layer produced by the last reduction is called the base layer.

The base layer is encoded. The transmitter then performs prediction where possible, encoding those pixels which cannot be predicted, to transmit the next higher resolution layer, using the base layer as a reference. This higher-resolution layer is then used as a reference to predict and encode a still higher-resolution layer. This progression is continued until the original image has been transmitted.

2.2.2 Sequential Transmission

Sequential transmission consists of dividing the original image into horizontal stripes and then transmitting each stripe in the progressive mode. The whole series of progressive transmissions described above is performed on each stripe before it is begun on the next.

2.3 The JBIG Base System

2.3.1 Summary

The base system consists of two major parts: image reduction and transmission. The base specification [1] recommends the PRES image reduction algorithm [2] (Progressive Reduction Scheme for Bi-level Images), but does not require it. The following description includes PRES, and parts of the rest of the system, especially deterministic prediction (DP), depend upon the employment of PRES.

In the progressive transmission mode, the encoder reduces the whole image a specified number of times, e.g. 5. The highest image layer, R0,[2] is

[2] The notation used here is not necessarily a part of the JBIG Base System Specification.

reduced to form layer R1, R1 to form R2, ..., R4 to form R5. R5 is called the base layer; R0 through R4 are called difference layers. One of the study objectives was to measure the total compression as a function of how many reductions are performed, including none.

Transmission is begun by the encoder's encoding, and the decoder's decoding, the base layer, using a binary arithmetic coder with contexts consisting of causal pixels (pixels known to both encoder and decoder) in the neighborhood of the target pixel. AT (Adaptive Template) is employed: the context template has a floating pixel that can change if a candidate floating pixel is a much better predictor. Since the base layer has no reference layer, no prediction of any kind is attempted except in the sense that the contexts act as predictors to improve data compression.

Next, layer R4 is transmitted. Because a reference layer (R5) is now available, prediction is employed. Both TP (typical prediction) and DP (deterministic prediction) are employed, with TP tried first. If a pixel in layer R4 is TP, it is not encoded. If not, the pixel is tested for being DP, and if so, is not encoded. If the pixel is neither TP nor DP, it is encoded with adaptive context templates which include pixels from both layers. The floating pixel is always in the layer being encoded.

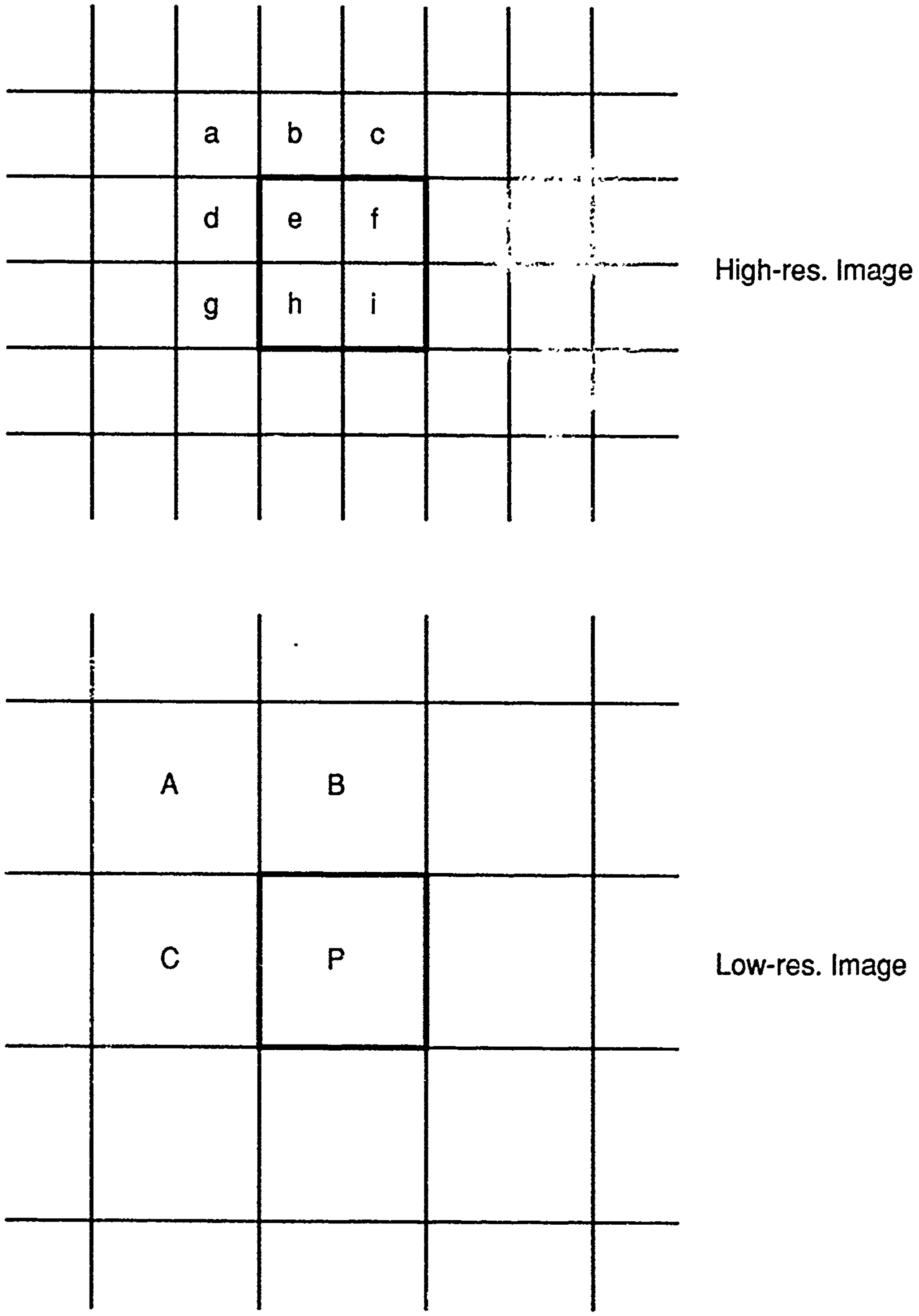
Similarly, layers R3, R2, R1 and finally R0 are encoded and decoded, with the next lower-resolution layer (next higher layer number) serving as the reference layer.

JBIG investigators considered two types of scan: row scan and z scan,[3] and they selected row scan.[4] In row scan, each image line is transmitted in its entirety before the next is begun. A prediction or context template may contain pixels anywhere above the current line or to the left of the target pixel in that line. In z-scan, two lines are scanned simultaneously. At each point in the scan, a square block of four pixels, two in each of the two lines, is transmitted, and the scan is advanced to the next such block. Any pixel above the pair of lines or to the left of the block in either line of the pair can belong to templates.

In sequential transmission, the progressive mode is applied to successive horizontal stripes of the original image. Each stripe is reduced the prescribed number of times, and the base layer and then successively higher-resolution layers are transmitted. The whole process is performed on each stripe, which is treated as a small image being transmitted in the progressive mode. Some proposals suggest using the first line of the next stripe as the assumed line below the bottom of the current stripe instead of a replication of the last line of the current stripe. These boundary conditions are described below for the progressive case. There is also the possibility of saving the binary arithmetic coder contexts after encoding (decoding) each layer of a stripe, so they can be used again for the next stripe. This might improve compression because the binary arithmetic coder would not have to readapt to the image statistics after initially assuming equal black and white probabilities in all contexts.

2.3.2 Image Pixel Notation

The following description applies to both image reduction and transmission. Figure 2.1 shows the relationships among, and notations for, pixels in an image layer having a given resolution and in the layer having half this resolution. In what follows, the former layer will be called the high-resolution layer and the latter the low-resolution layer. High-resolution pixels e, f, h and i register with low-resolution pixel P, as shown by the heavy



High-res. pixels e, f, h and i correspond to low-res. pixel P.

Figure 2.1 High and Low Resolution Pixel Notation and Registration

rectangles.

In image reduction, pixel values in the high-resolution layer, and those in the low-resolution layer already determined, are used to determine the next low-resolution layer pixel. In image encoding and decoding, pixel values in the low-resolution layer and those already encoded (decoded) in the high-resolution layer are used to try to predict the next high-resolution pixel.

In JBIG literature and in this report, the term "phase" is sometimes used to refer to one of the four high-resolution pixels denoted by e, f, h and i. Phase 0 refers to e, 1 to f, 2 to h and 3 to i.

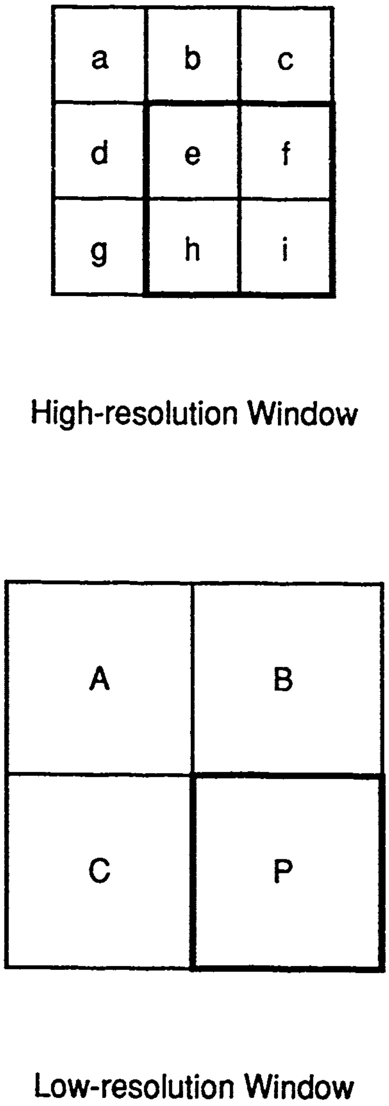
2.3.3 PRES Image Reduction

The simplest method of reducing an image to half its size in both dimensions is straight sub-sampling: keep every other pixel in a given high-resolution line, and do this to every other line. In bi-level images, however, this method quickly washes out detail. In images containing text or line drawings, lines forming the drawings or text characters grow thinner with each reduction, and soon disappear. In half-tone images, gray levels become badly distorted.

JBIG participants selected PRES from a number of image reduction algorithms, for example: Progressive Coding Scheme Using Block Reduction (PCSB)[5]; rogressive Edge Decomposition of Facsimile Images (PED)[6]; and a projection method that combines filtering, sub-sampling and line preservation[7].

The PRES algorithm consists of two parts: (1) a formula for determining a low-resolution pixel value, and (2) a list of exceptions that override the formula. The formula is, in effect, a filter, and the exceptions seek to preserve such features as lines, edges and dither patterns.

Figure 2.2 shows the windows that are scanned over the high- and low-resolution images as reduction is performed. (The low-resolution image can be



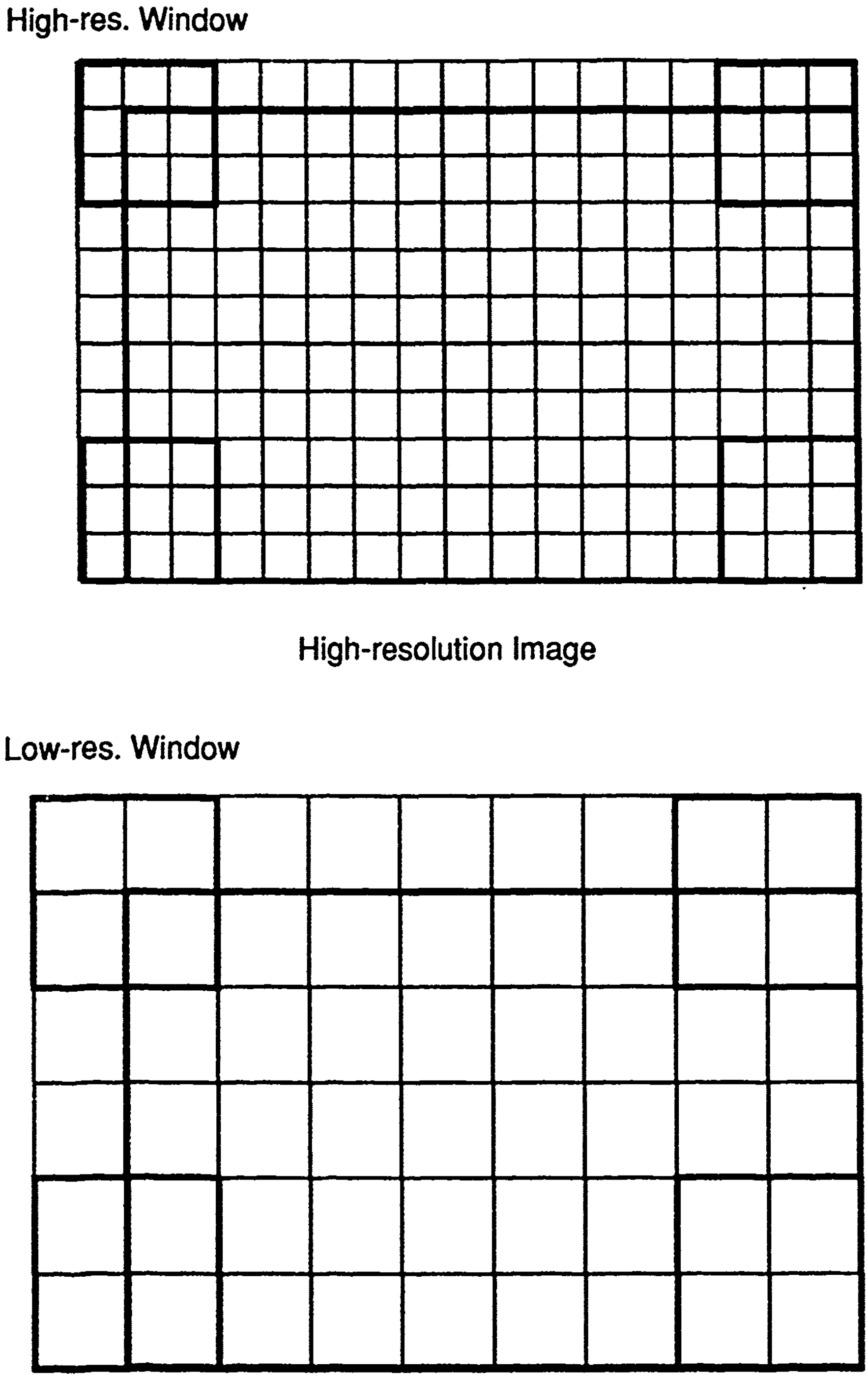
a	b	c
d	e	f
g	h	i

High-resolution Window

A	B
C	P

Low-resolution Window

Figure 2.2 High and Low Resolution Windows



High-res. Window

High-resolution Image

Low-res. Window

Low-resolution Image

Note: The high-resolution image dimensions are assumed to be even. See text for description of treatment when either dimension is odd.

Figure 2.3 Window Alignment at Image Corners

thought of as containing "blank" pixels, with PRES filling in the blanks.) The heavy rectangles in the two windows show how the four high-resolution pixels e, f, h and i overlay pixel P, the low-resolution pixel to be determined at a particular place in the scan. Low-resolution pixels A, B and C have already been determined.

Figure 2.3 shows how the high- and low-resolution windows align with the image corners, where the high-resolution image is assumed to have an even number of pixels per line and an even number of lines. The large, heavy rectangles represent the actual image boundaries. Imaginary white pixels are assumed above and to the left of both images. In the high-resolution image, if the number of pixels per line is odd, each image line is assumed to have a white pixel appended to its right end, therefore making the effective image width even. Similarly, if the number of lines is odd, an extra line of white pixels is assumed to be appended.

The width and height of the low-resolution image is given by:

$$W = (w + 1)/2$$

and

$$H = (h + 1)/2,$$

where W and H are the low-resolution image width and height, w and h are the actual high-resolution width and height, and the divisions are <u>integer</u> divisions, the results equivalent to rounding the real divisions w/2 and h/2 to the nearest integer. Thus, for odd w (h), W (H) is half the width (height) of the high-resolution image with the white pixel (line of pixels) appended.

High-resolution pixels a, b, c, d and g, and low-resolution pixels A, B and C, always have known values, because imaginary white pixels are assumed above and to the left of both images. Thus, the PRES algorithm need not make exceptions at the image boundaries; assuming that the images are framed by white pixelstakes care of the boundary conditions and of the case of odd high-resolution image width or height.

After the first low-resolution pixel has been generated, the high-resolution window is moved two pixels to the right, and the low-resolution window one pixel to the right. The first line of each image is thus scanned until the end-of-line condition is reached as shown in Figure 2.3. Next, the high-resolution window is moved to its starting position, but down two lines. Similarly, the low-resolution window is moved to its starting position, but down one line. This scanning is continued until all low-resolution pixels have been generated.

The PRES reduction formula is, in the absence of an exception:

Set SUM = $4e + 2(b + d + f + h) + a + c + g + i - 3(B + C) - A$.

If SUM is greater than or equal to 5, set $P = 1$, otherwise 0.

This formula is, in effect, a filter, with previously-determined low-resolution pixels included.

The exceptions reside in a table containing exception states and exception pixel values. An exception state is formed by concatenating pixel values A, B, C, a, b, c, d, e, f, g, h, and i in left to right order and right-justifying these bits in an integer word, where these pixel values represent some feature that must be preserved. The existence of an exception is tested by similarly assembling the actual pixel values into a state and searching the exception table for a matching state. If a match is found, the required value of P is taken from the exception table; otherwise it is computed from the formula.

In an actual PRES reduction implementation, the whole algorithm, including the formula and the exceptions, is contained in a PRES look-up table to save processing time. For each low-resolution pixel, P, to be generated, a state is assembled as described above. This state is used as an index to the table, from which the required value of P is retrieved. The PRES table is originally generated by executing the actual PRES algorithm once for each of the 4096 possible state values.

2.3.4 Generic Typical Prediction (TP-G)

JBIG participants selected generic typical prediction from among other schemes[8] that included rules tied to the image reduction rules in addition to the generic rules to be described here. TP-G refers to TP based only on the generic rules. For the remainder of this report, the notation TP implies TP-G.

In the following description, pixel notation and the registration of the high- and low-resolution images are as shown in Figure 2.1.

In generic typical prediction, a TP cluster is defined as a low-resolution pixel surrounded by a neighborhood of pixels of the same color. The neighborhood consists of pixels horizontally, vertically and diagonally adjacent to the pixel in question. (The boundary conditions for TP-G and for all encoding and decoding operations are the same as for image reduction, except that the imaginary pixels below an actual image are assumed to be copies of the pixels in the last image line, instead of white.)

If low-resolution pixel P belongs to such a cluster, then high-resolution pixels e, f, h and i almost always have the same color as P. This is particularly true in images having large areas of solid color, as in line drawings and text.

Prior to encoding a pair of high-resolution image lines, the encoder performs a TP test consisting of examining the corresponding low-resolution line. (No prediction of any kind is employed for the base layer, where there is no lower-resolution layer.) For each low-resolution pixel, P, belonging to a TP cluster, high-resolution pixels e, f, h and i are examined. If they all have the same color as P, then the four high-resolution pixels are predictable. If not, then a TP exception has been encountered.

In the JBIG Base System, if a TP exception is found anywhere in the low-resolution line, then that whole line is declared as an exception, and no TP is attempted in the corresponding pair of high-resolution lines. After the TP exception check has been completed, an exception bit, e.g. 1 for exception, 0 for no exception, is encoded in its own context to tell the decoder whether to employ TP while decoding the pair of high-resolution lines.

Other schemes have been proposed for reporting TP exceptions. For example, one could encode an exception bit more than once per line, or one could encode a pointer to the next exception. The selected method gives good results, and is computationally simple. This method is described in, for example, JBIG Document N131.[9]

If no exception is found, then, whenever a low-resolution pixel, P, belonging to a TP cluster is encountered, the corresponding pixels, e, f, h and i, are not encoded. The decoder, having been told that there is no TP exception, merely inserts pixels having the same color as that of pixel P.

Since the same low-resolution line applies to two high-resolution lines, the study system saves execution time by having the TP test routine generate a string of TP flag bits, one per low-resolution pixel.[3] A bit value of 1 means that the low-resolution pixel belongs to a TP cluster. Then, as each high-resolution line of the pair is scanned, two pixels at a time, the flag bits are also scanned,

[3] This method was suggested by C Chamzas of AT&T in a private communication.

one bit at a time. If the flag bit is 1, then the two high-resolution pixels are not encoded.[4]

2.3.5 Deterministic Prediction (DP)

In the JBIG Base System, DP is tied to the PRES reduction rules.[10] The encoder and decoder determine whether a target difference layer pixel is DP by a table search. The table is created from a set of DP rules specified in JBIG documents N141 and N199.[11] A DP state is constructed from a DP template consisting of pixels from the lower-resolution reference layer as well as causal pixels from the higher-resolution difference layer being encoded (decoded). The table is then searched for a matching state for the phase in question (0, 1, 2 or 3, i.e., pixels e, f, h and i, as in Figure 2.1). If a match is found, then the target pixel value is taken from the table instead of being encoded or decoded.

The DP states are:

Phase	State (Predictor Pixels)	Target Pixel
0	A,B,C,P,a,b,c,d	e
1	A,B,C,P,a,b,c,d,e	f
2	A,B,C,P,a,b,c,d,e,f,g	h
3	A,B,C,P,a,b,c,d,e,f,g,h	i

As in PRES, computation time is saved by using one look-up table for all the states of all the phases. For a given phase, the DP state is assembled from the appropriate pixels and a table look-up is performed. There is always a table

[4] The base system employs row scan; each high-resolution line is scanned completely before the next is begun. An alternative is z-scan, in which two lines are scanned simultaneously, and pixels e, f, h and i are processed together.

entry, whether or not the target pixel is DP. If it is, the table delivers the predicted value, 1 or 0; otherwise it delivers a "don't predict" code, e.g. 2 or 3.

The encoder simulator in the study system verifies correct prediction. Any prediction error would imply a bad PRES or DP table or a program failure.

2.3.6 Adaptive Context Templates (AT)

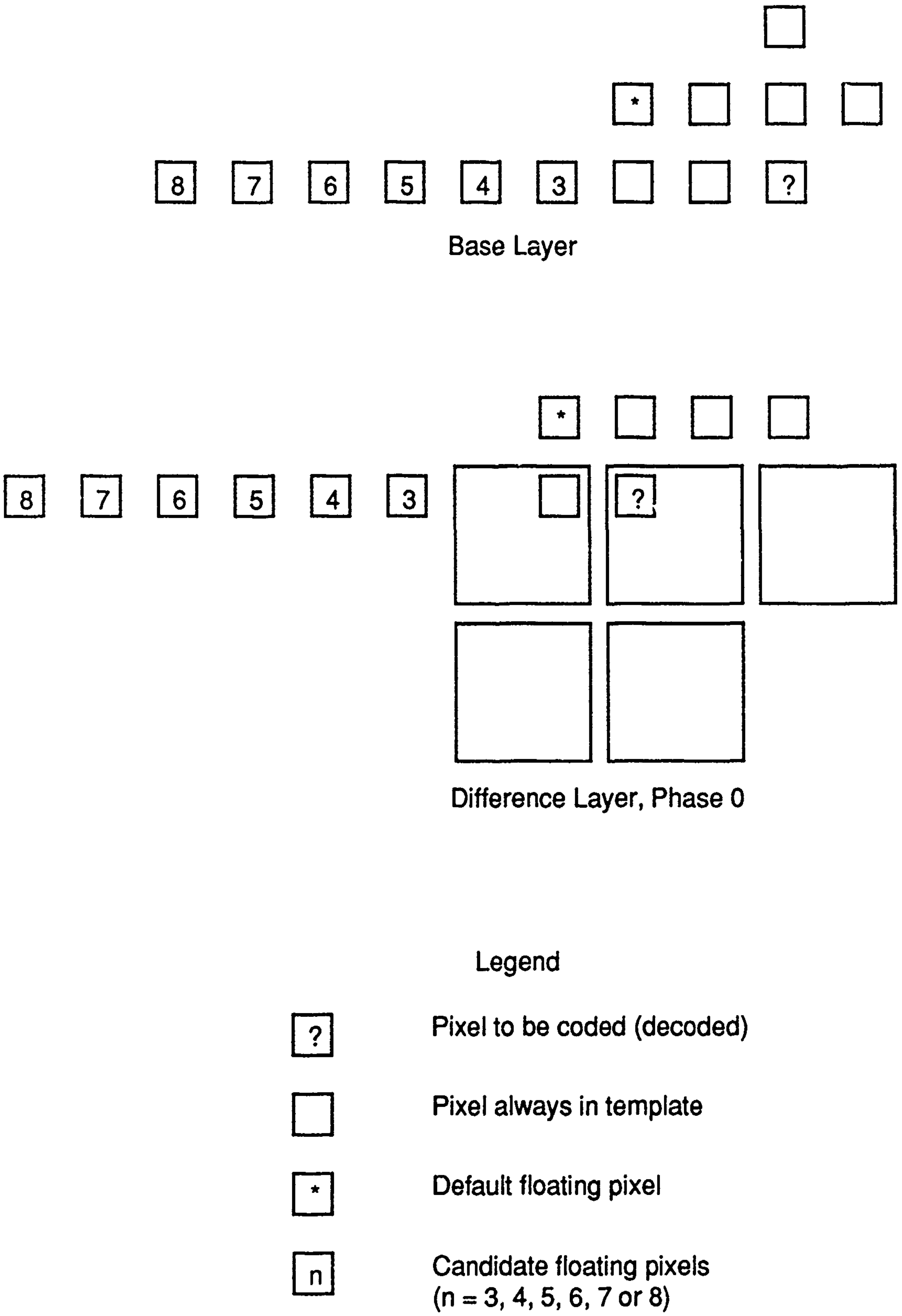
2.3.6.1 Template Descriptions

Figure 2.4 (two pages) shows the context templates for encoding (decoding) base layer pixels and difference layer pixels e, f, h and i (phases 0, 1, 2 and 3). These templates are documented in JBIG Document N174.[12]

Small squares represent high-resolution pixels for the difference layers, and all base layer pixels. Large squares represent low-resolution (reference) layer pixels for difference layer processing. All pixel positions are shown relative to the pixel being encoded (decoded), which is labeled with a question mark.

Squares not containing characters represent pixels that are always in the template. In each diagram there is one square labeled with an asterisk (*). This represents the default AT floating pixel; it is initially in the template, but will not be following an AT swap unless it is swapped back in later. Squares labeled 3, 4, 5, 6, 7 and 8 denote candidate floating pixels, not initially in the template. These are called "lag" pixels because they lag behind the encoded (decoded) pixel by a number of pixels equal to their label values.

Whether or not the default or one of the lag pixels is the floating pixel, the base layer has 7 pixels in its template. Thus, there are 128 contexts for the base layer, i.e. 128 possible combinations of black and white pixel values in the template. For each difference layer template, there are 10 pixels, some high-



Base Layer

Difference Layer, Phase 0

Legend

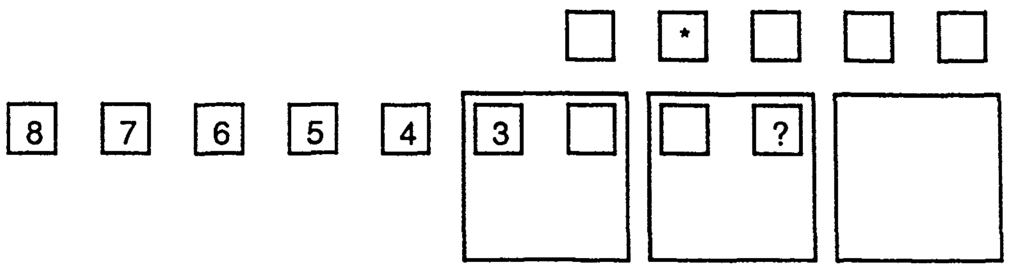
? Pixel to be coded (decoded)

□ Pixel always in template

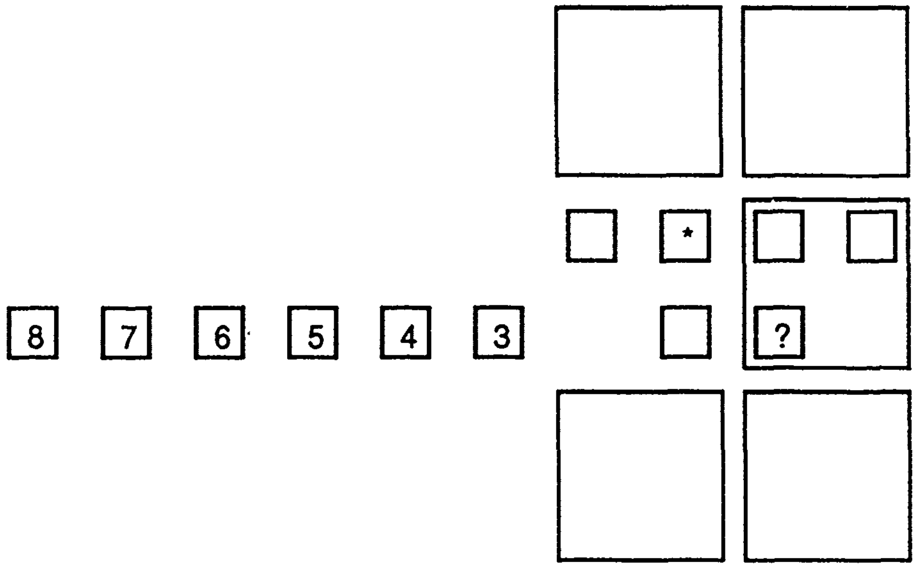
* Default floating pixel

n Candidate floating pixels
(n = 3, 4, 5, 6, 7 or 8)

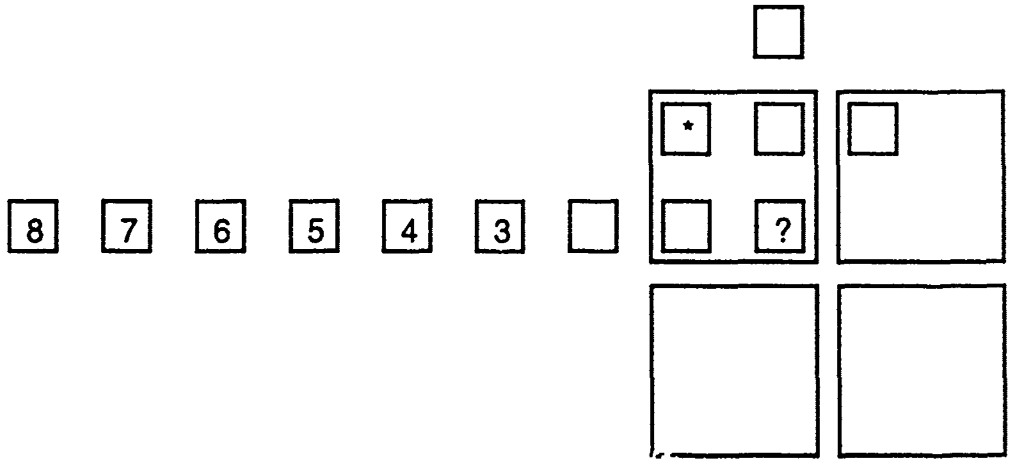
Figure 2.4 Adaptive Context Templates (Part 1 of 2)



Difference Layer, Phase 1



Difference Layer, Phase 2



Difference Layer, Phase 3

Figure 2.4 (Continued) Adaptive Context Templates (Part 2 of 2) and some low-resolution. Thus, there are 1024 contexts for each phase, or a total of 4096 contexts for difference-layer processing. (The extra context for TP exceptions is not included in this total.) Note that it is perfectly legitimate for low-resolution pixels to lie below or to the right of the encoded (decoded) pixel; all low-resolution pixels are known to both encoder and decoder.

2.3.6.2 AT Algorithm

The AT algorithm (Adaptive Template, formerly called AC for Adaptive Context) is described, for example, in JBIG Documents N139-R2[13] and N208-R0[14].

The default floating pixel and the six lag pixels, shown in each diagram of Figure 2.4, are collectively called AT (formerly AC) pixels; at any given time, one of them is the floating pixel. The difference layer templates are so designed that the seven AT pixels have the same spatial relationships to the target (encoded or decoded) pixel, independent of phase. Therefore, floating pixel swaps can be tested and performed for all phases at once. (Document JBIG N139 does not assume phase independence; N208 does.)

Floating pixel changes are controlled in the following manner. To each AT pixel is allocated a counter whose count is initially zero. A "total" counter, C_{all}, is maintained and also initialized to 0. For each target pixel, a decision is made whether to update the counters. Criteria for this decision are given later. Updating the counters consists of incrementing the total count, C_{all}, and each counter associated with an AT pixel having the <u>same color</u> as the target pixel. These updates are made, of course, after the target pixel has been transmitted (received); the decoder performs the same updates.

At appropriate times (criteria also given later), decisions are made whether to swap the current floating pixel with another AT pixel. The decision criteria are designed to make AT switching infrequent, because, after the switch, the probability estimates become poor, with a consequent degradation of compression, until the binary arithmetic coder probability estimator has time to adapt to the change.

AT counts are updated only if all of the following conditions hold: (1) The current coding (decoding) line is not the first or second (difference layer), or not the first (base layer); (2) All lag pixels are inside the actual image; and (3) The current pixel is not typically predictable. TP pixels are avoided because they tend to occur in regions of solid color, where the use of AT is not intended. (DP pixels are included, even though they are not encoded.)

The swap/no-swap decision criteria are taken directly from JBIG N139-R2. Let C_{max} be the maximum value of the counts in the seven AT counters, and C_{min} be the minimum count. Let E_{max} and E_{min} be the maximum and minimum counts excluding the count for the default floating pixel. Let C_{float} be the count for the current floating pixel, default or not. A floating pixel swap is made only if all of the following conditions are met:

1. $C_{max} > (7/8)\ C_{all}$ (The study system tests for $8C_{max} > 7C_{all}$)
2. $C_{max} - C_{float} > C_{all} - C_{max}$,
3. $C_{max} - C_{float} > C_{all}/16$,
4. $C_{max} - (C_{all} - C_{float}) > C_{all} - C_{max}$,
5. $C_{max} - (C_{all} - C_{float}) > C_{all}/16$
6. $C_{max} - C_{min} > C_{all}/4$,
7. Current floating pixel is not the default or $E_{max} - E_{min} > C_{all}/8$.

Condition 1 requires that the count maximum be very strong. AT switching is desired in dithered images and nowhere e.se. Condition 2 requires that the maximum count is moving more than half way rrom the current floating pixel count to the maximum possible count. Condition 3 makes sure that the maximum count is considerably greater than that of the current floating pixel. Conditions 4 and 5 are similar to 2 and 3, but prohibit switching when the current floating pixel has strong negative correlation with the target pixel. Condition 6 requires that not all counts are high; AT swaps are not wanted in regions of solid color.[5] Finally, condition 7 makes it hard to swap the initial default floating pixel.

The test for swapping is made at the end of every second line for difference layers, at the end of each line for the base layer, provided that the total count, C_{all}, is at least 2048. If the test is made, all counters are reset to zero, whether or not a swap occurs.

2.3.7 Binary Arithmetic Coder

2.3.7.1 Coder Selection Criteria

The question of which binary arithmetic coder (BAC) would be recommended was an outstanding JBIG issue until the late winter of 1989-1990. The selected coder was to have the ge..eric name GABAC (General Adaptive Binary Arithmetic Coder) and the specific name QM Coder, where QM stands for Quick-Minimax, Quick-Melcode or Q-Modified.[15] This coder was to combine the best features of the three strongest contenders: Q-coder (IBM), Mel-Coder (Mitsubishi) and Minimax coder (AT&T).

[5] In difference layers, such regions tend to be comprised of TP pixels, which are not included in the AT counts. In the base layer, however, there is no prediction.

JBIG document N158-R0[16] gives an excellent review of the selection criteria, which are summarized below. Appendix A of this report is a description of the principles of a BAC.

MPS/LPS

It was agreed by all parties that classifying a binary symbol as MPS (more probable symbol) or LPS (less probable symbol) is superior to high/low, 1/0, black/white, etc..

Computing MPS/LPS Sub-interval

The better the MPS and LPS probability estimates, the better the compression, other factors being equal. In Appendix A the BAC description specifies multiplication to compute the LPS sub-interval. Full multiplication is costly, whether implemented in hardware or software. Approximations to the multiplications vary among the different coders; all lead to some compression degradation with respect to full multiplication, typically in the order of tenths of a percent to a percent.

Carry Management

Means must be provided to prevent the propagation of carry bits into data that have already been transmitted. One of the methods, bit stuffing, has the beneficial effect of creating a range of values for the byte following a byte of all 1's that can never occur in the code stream. An all-1's byte followed by an "illegal" byte can therefore be used as a marker code, as in the JPEG (Joint Photographic Experts Group) environment.

Byte vs. Bit Output

As of October 1989, the Q-coder was the only contender whose output was byte oriented. This feature simplifies the output path and provides a means of transmitting marker codes.

Skew

The greater the MPS/LPS probability skew, the greater the compression. The greater the precision of the probability estimator, the greater the skew it can support. (The probability estimator must never estimate an LPS probability of zero, else the LPS sub-interval would vanish, and the code stream could not be decoded.)

Probability Estimation

The Q-coder probability estimator is much simpler than those of the others, because the probability estimate is updated without counting symbols when the A interval is renormalized. The Q-coder estimator behaves like a "finite state machine" represented by a table described later. The other two coders count symbols, with varying means of compromising between memory allocation and probability estimation accuracy.

Fast Learning ("Fast Attack")

The reference covers fast learning under probability estimation. It is brought out separately in this report because of its great importance in the JBIG environment, where there are thousands of contexts in difference layer processing. Compression in any given context is good when both of the following conditions are met: (1) at least one of the context template pixels is a good predictor, and (2) the probability estimate for each context is accurate. In the JBIG Base System, the probability estimates of all contexts are initialized to 0.5; hence compression does not begin to get good until the estimates have had time to approach the actual probabilities. "Fast attack" refers to the ability to estimate probabilities approaching the true values after only a few "hits" on any given context.

2.3.7.2 The Selected Coder

The coder finally selected is an enhanced version of the Q-coder whose probability estimator table supports fast attack. This enhancement is described in JBIG Document N213.[17] The document authors represent Mitsubishi, IBM and AT&T, originators of the three contenders. The history and properties of this coder are summarized below.

In the original Q-coder, invented by IBM,[18] the LPS probability estimates had 12 bits of precision. The coder approximated the products A*Q and A*P by Q and A-Q respectively, where A is the probability interval, and P and Q are the estimated MPS and LPS probabilities. See Appendix A for BAC principles and notation.

For the multiplication approximations to be reasonably accurate, the value of A must be kept near 1. The Q-coder therefore renormalized by doubling both A and C whenever the value of A fell below 0.75. (The actual arithmetic was performed on scaled integers with implied binary points.) Since Q is always 0.5 or less, renormalization always occurred after the encoder or decoder encountered an LPS; it occurred occasionally when an MPS was encountered.

The probability estimator took the form of a finite state machine. Such a machine has the property that its next state depends upon its present state and its present inputs. In the original Q-coder, the present state (for each context) combined (1) an index to a table of estimated LPS probabilities, and (2) the current value of the MPS. Each table record contained, not only the LPS probability estimate for that state, but also two "next state" (next index) values, one for when the encoded (decoded) symbol was the MPS and one for the LPS. When the state was such that the MPS probabilities were both (very near) 0.5

and an LPS was encountered, then, instead of moving to a new table index, the estimator simply switched the value of the MPS.

The memory allocated to each context contained the current MPS value and the table index for that context. Whenever the encoder (decoder) renormalized (doubled A and C and delivered (acquired) a byte of coded data if a new byte was ready), the context was updated with a new table index or a new value of the MPS.

The JPEG (not JBIG) BAC[19] is an extension of the original Q-coder. Its LPS probability estimates have been extended to 15 bits of precision to support greater skew. The coder also provides for swapping the MPS and LPS subintervals if the former becomes smaller than the latter. This can happen because of the multiplication approximations when A is at or near its minimum allowed value of 0.75, and the estimated LPS probability, Q, is near 0.5. Since this swapping can occur only during renormalization, it adds little computational burden in the MPS path which, by definition, is more probable than the LPS path. Detailed flow diagrams of the JPEG coder are given in the reference.

JBIG document N213-R0, reference cited above, describes the fast-attack enhancement of this JPEG coder. Two "fast-attack ladders" are added to the main "state machine." The first ladder is entered initially. The state transfers to the second ladder when the first LPS occurs, or a series of MPS's carry the state off the ladder.[6] The state transfers from the second ladder to the main state machine when a second LPS occurs or a series of MPS's carries the state off that ladder. Once the state enters the main state machine, it remains there

[6] If the very first symbol in a given context is the LPS, the state remains the same except for a switch of the MPS value.

for the balance of the encoding (decoding) task. The fast-attack ladders serve to bring the probability estimates for any given context into the "right ball park" with very few "hits" on that context.

As this report was being written, a new revision of the JPEG specification was released[20] This describes a further refinement of the O-coder, including the fast-attack ladders. Carries are handled without bit stuffing, but all-ones bytes are followed by all-zeros bytes to allow for marker codes. It is anticipated that the ultimate JBIG and JPEG standards will specify the same binary arithmetic coder.

3.0 THE STUDY SYSTEM

3.1 Overview

The study system simulates the JBIG Base System as defined in March, 1990, with the exception that the study system simulates the progressive transmission mode only. The insights gained and conclusions drawn from the simulations would have been negligibly different had both progressive and sequential transmissions been simulated.

The documents on which the major components of the study system are based are:

PRES	JBIG Document N198
TP	JBIG Documents N130 and N131
DP	JBIG Documents N199 and N141
Context templates	JBIG Document N174
AT algorithm	JBIG Documents N139-R2 and N208-R0
BAC	Document X3L2.8/90 - 004 and JBIG N213-R0

The study system was written, not merely to repeat simulations performed by others, but to obtain additional data that may influence the final standard. These data are specifically: (1) the optimum number of image reductions, sometimes none, to maximize overall compression, and (2) the contributions of the major system components, namely TP, DP and binary arithmetic coding (including AT), to data compression.

Recent proposals flowing from the ongoing JBIG investigations, but not implemented in the study system, include:

Starting Layer Algorithm with Optimal Templates

Described in JBIG Document N190 Rev. 1[21], the Starting Layer Algorithm transmits a given image layer directly, without the benefit of a reference layer, without TP or DP, and with optimal context templates different from those specified in JBIG Document N174. The study system has the option of simulating this algorithm by treating any image layer as the base layer, but employs the base layer template described in JBIG document N174. Because the Starting Layer Algorithm does not employ any prediction, it is much simpler than the general algorithm, and is a serious contender for the JBIG standard.

At In Transmitter Only

JBIG Document N204-R0, reference cited, specifies that AT processing be done only in the transmitter. Template changes are transmitted when necessary; they are so infrequent that the overhead is negligible.

Pure Binary Arithmetic Coder Output

The JPEG/Fast-Attack coder, described in JBIG Document N213-R0 and JPEG Document X3L2/90 - 004 (references cited), manages carries by bit stuffing. "Pure" arithmetic coder output refers to a bit stream that exactly represents the binary fraction, C, to the precision required for decoding. Pure output relieves the decoder of the need to test for and process stuff bits or other special data required to cope with the carry problem.

JBIG document N219[22] proposes a method of generating pure output. It consists of counting consecutive bytes of all 1's instead of delivering them to the bit stream. When a byte that is not all 1's is encountered, the previous such byte is still in the output buffer, but has not yet been transmitted.

If it is determined that a carry is being propagated into a string of all-1 bytes, it is added to the byte preceding the all-1 bytes, and then as many bytes of all 0's are output as there were all-1 bytes. If no carry is being so propagated, then the counted number of all-1 bytes are output. In either case, after the required number of bytes are output, the counter is reset to 0.

A 32-bit counter would never overflow with an image of any practical size, even if all the bytes were all 1's, the probability of which is minuscule. The reference states that the all-1's byte count never exceeded 4 for any image processed.

The pure output is implemented in the May 1990 JPEG specification (JPEG-8-R5.2, reference cited), but each byte of all ones is followed by one of all zeros to allow the insertion of marker codes into the code stream. This is called byte stuffing. In an environment that never uses marker codes, byte stuffing would be unnecessary.

The study system consists of three program: PRES2, JBENCODE and JBDECODE. The system produces compression statistics and counts the numbers of TP and DP pixels in difference layers to determine the contributions of the system components to data compression.

During development and testing, each program produced intermediate printouts so that the innermost workings of the algorithm could be monitored. Special small test "images" were contrived to test all aspects of every part of the system. The intermediate printouts were, of course, disabled during production simulations.

3.2 Programs

3.2.1 Program PRES2

Purpose

To reduce an image to half its size in both directions. If either dimension is odd, the reduced image dimension is rounded up. The name PRES2 arises because the original PRES program performed the actual computations and exception searches; PRES2 employs a look-up table. PRES2 is executed repeatedly to produce images of successively lower resolution, each execution performing one reduction.

Inputs

PRES look-up table file name
File name of image to be reduced
Reduced image file name
Width and height in pixels of image to be reduced

Outputs

Reduced image
Printout of the reduced image dimensions

3.2.2 Program JBENCODE

Purpose

To simulate the transmission of one image layer, difference or base.

Inputs

File name of image layer being encoded

Base or difference layer option

File name of reference layer, not applicable to base layer

Prediction option, not applicable to base layer

DP rules file name if DP is employed

Compressed data file name

Encoder statistics file name (print file)

Option to enable AT

Width and height (pixels) of image being encoded

Width and height of reference layer, not applicable to base layer

The prediction option is TP, DP or both (TP tried first), and is not applicable to the base layer. The AT option is valid for base or difference layer encoding. The JBIG Base System is simulated by invoking both TP and DP and by enabling AT.

Outputs

Compressed data file

Statistics print file

The statistics print file gives the simulation results. In addition to the compressed data bit count, it contains other data designed to give insight into what the JBIG algorithm does. These statistics are described in Section 4.3.

3.2.3 Program JBDECODE

Purpose

To simulate the reception of one image layer, difference or base.

Inputs

The inputs are the same as for JBENCODE except that the first input is the name of the file where the decoded image is to be written, and the statistics file name specifies where the decoder statistics are to be placed. The options must, of course, match those invoked in the encoder simulation to ensure accurate decoding of the image.

Outputs

Decoded image

Decoder statistics

The decoder statistics are much less elaborate than those produced by the encoder, since the decoder duplicates what the encoder does. The decoder statistics were designed merely to check the software.

Because the JBIG algorithm produces lossless image transmission, the only purpose of simulating the decoder was to verify the software integrity of the encoder. During development and testing, the decoder was always simulated. In production simulations the decoder was simulated only occasionally to give spot checks.

4.0 SIMULATIONS

4.1 Simulation Objectives

The simulation objectives were: (1) to prove that the study system simulates the JBIG Base System algorithm, and (2) to accumulate and document data not published in any JBIG literature. Specifically, these data are: (1) the total compression as a function of the number of reductions, and (2) the data compression contributed by the major parts of the algorithm (TP, DP, the binary arithmetic coder, etc.) during difference layer processing. These data could have considerable influence on the compromise between data compression and system complexity.

4.2 Test Images

The test images are referred to as the Stockholm images in most JBIG documents. Table 4.1 summarizes these images.[23] The starting image resolutions are 400 dots per inch; five PRES2 reductions produce images having resolutions of 200, 100, 50, 25 and 12.5 dots per inch.[7]

4.3 Encoder Simulation Statistics

The encoder simulator, JBENCODE, reports the following statistics:

- Number of compressed data bits
- Number of TP pixels (difference layers only)
- Number of DP pixels (difference layers only)
- Number of bits sent to the binary arithmetic coder
- Number of TP exception lines (if TP is invoked)
- Number of floating pixel changes (if AT is invoked)

The encoder simulator reports the image line number where each floating pixel change occurs.

To give insight into the performance of the binary arithmetic coder in the JBIG environment, the encoder simulator periodically produces: (1) a histogram showing the number of contexts lying in various ranges of number of "hits,"

[7] Actually, PRES2 halves the number of pixels per line and number of lines. The "resolutions" would be those observed were the reduced images enlarged to the starting size.

No.	Type	Dimensions width x height	Source
1	Letter type	3072 x 4352	Eastman Kodak USA
2	Roman type and figures	3072 x 4352	Eastman Kodak USA
3	Japanese news paper	3072 x 4352	Eastman Kodak Japan
4a	Half tone image 8x8 dither	3072 x 2048	Eastman Kodak USA
4b	Half tone image ERR dif.	3072 x 2048	Eastman Kodak USA
4c	Half tone image 4x4 dither	3072 x 2048	Eastman Kodak USA
4d	Half tone image 3x3 dither	3072 x 2048	Eastman Kodak USA
5	Hand writing	3072 x 4352	Eastman Kodak Japan & USA
6	Mixed document	3072 x 4352	Eastman Kodak USA
7	Line drawing scanned	3072 x 4352	Eastman Kodak USA
8	Line drawing generated	3072 x 3040	Delta Information Systems USA
9	Error diffused image	1024 x 1024	Eastman Kodak USA
10	Computer generated line	4096 x 5856	Delta Information Systems USA

Table 4.1 Test Image Descriptions

and (2) a weighted average of the context indices into the probability estimator table. The higher the index for any given context, the greater the MPS/LPS probability skew, and hence the better the compression contributed by that context. The weighting factor is the number of "hits" on the context. While contexts with very few hits give less compression than those with more (because their probability estimates are less accurate), they contribute to the bit stream less often, and so the overall compression degradation is slight.

4.4 Procedure

The following steps were performed for each test image:

1. Denote the original image as layer R0 (zero reductions)
2. Execute PRES2 5 times to produce layers R1, R2, R3, R4 and R5
3. Execute JBENCODE to encode all 6 layers as if they were base layers
4. Execute JBENCODE to encode layers R0 through R4 as difference layers

A few layers of a few images were decoded to check the encoder software integrity.

All production simulations were performed with the system parameters set to simulate the JBIG base system. However, in a few tests to evaluate the contribution of AT to data compression, all prediction was disabled, and separate simulations were performed with AT enabled and disabled. The results of all simulations, including these specisl tests, are presented in the next section.

5.0 RESULTS

5.1 General Discussion

5.1.1 Comparison of Study and JBIG Base Systems

To prove that the study system simulates the JBIG Base System algorithm, a comparison was made between simulation results reported in JBIG Document N213-R0 (reference cited) and those obtained with the study system. These will be called the reference and study results respectively.

The reference results were taken from Table 3A of the reference document. The simulation parameters were: PRES + TP + DP + AC (now AT), without stripes (i.e. progressive transmission), with minimum termination (of the binary arithmetic coder), and without special codes. The reference and study simulators employ the JPEG/FA coder as specified in JPEG document X3L2.8/90 - 004, with the enhancement for fast attack added as described in JBIG Document N213-R0, both references cited.

Both sets of data apply to full progressive encoding of the original 400 dots-per-inch images, namely: (1) PRES reducing these images to produce images having resolutions of 200, 100, ..., 12.5 dots per inch, and (2) encoding the 12.5 dots-per-inch images as base layers and the remaining images as difference layers. The study and reference parameters are identical, except that in the study simulations the bytes required to produce an end-of-image marker code are included in the byte count.

Table 5.1 shows the results. Column 1 gives the image number, column 2 shows the total compressed data byte count cited by the reference for the entire progression, column 3 cites the total byte count reported by the study system, column 4 shows the difference between the study and reference byte counts (study minus reference) and column 5 gives this difference as a percentage relative to the reference byte count.

For all test images the results agree to within a few tens of bytes out of many thousands, and in some cases over 200 thousand bytes of compressed data. It is thus apparent that the study system, for practical purposes, exactly simulates the JBIG Base System algorithm as defined in March, 1990.

Image No.	Reference (bytes)	Study (bytes)	Difference (bytes)	Difference (percent)
1	13761	13771	+10	+0.073
2	16026	16041	+15	+0.094
3	149207	149219	+12	+0.008
4a	91344	91355	+11	+0.012
4b	113172	113163	-9	-0.008
4c	69258	69271	+13	+0.019
4d	113531	113541	+10	+0.009
5	33370	33384	+14	+0.042
6	228109	228080	-29	-0.013
7	24850	24865	+15	+0.060
8	7171	7191	+20	+0.279
9	99984	99996	+12	+0.012
10	17236	17246	+10	+0.058

Table 5.1 Comparison of Reference and Study Results

5.1.2 Simulation of Partial Progressions

Partial progressions were simulated to determine the optimum number of image reductions to minimize the total number of transmitted bytes. Layer numbers R0, R1, ..., R5 denote the number of reductions, 0, 1, ..., 5, required to produce those layers. Let $N_{b,i}$ be the number of compressed data bits required to transmit layer Ri as a base layer, and $N_{d,i}$ the number to transmit the same layer as a difference layer. (Layer R5 is always transmitted as a base layer, since there is no layer R6.) Then the total number of bits, T(k), required for a partial progression involving k image reductions is:

$T(0) = N_{b,0}$;

$T(k) = N_{b,k} + N_{d,k-1} + N_{d,k-2} + \ldots + N_{d,0} \; (k > 0)$.

The optimum number of reductions for minimum total compressed data is that k for which T(k) is minimum.

These tests included encoding the starting image as a base layer, as in the Starting Layer Algorithm described in JBIG Document 190, Rev. 1 (reference cited), except that the same base layer template (JBIG Document N174) was employed regardless of which image layer was treated as the base layer.

Table 5.2 summarizes the results of simulating these partial progressions. The table shows, for each image, the optimum number of reductions, including none, for minimum total compressed data, and the compression ratio (number of starting image pixels to total number of compressed data bits) achieved for the optimum number of reductions.

Data presented later show, for each image, the total number of compressed data bytes as a function of the number of reductions. The optimum numbers were 0 in the two error diffused images and in the 3x3 dithered image. In most of the remaining images the optima were 1; in a few, 2; and the optimum never exceeded 2.

5.1.3 Contributions to Data Compression

In base layer processing, data compression is produced solely by the binary arithmetic coder in concert with the adaptive context templates (AT). In difference layer processing, however, TP and DP also contribute significantly, although much of the contribution of TP and DP is "stolen" from that of the arithmetic coder. That is, if either TP or DP or both were not used, the

Image No.	Optimum Number of Reductions	Compression Ratio
1	1	112.2
2	1	111.7
3	2	11.5
4a	1	9.9
4b	0	9.6
4c	1	12.6
4d	0	11.0
5	2	51.4
6	1	7.9
7	2	69.7
8	1(*)	180.2
9	0	1.6
10	2	177.5

(*) Tied with 2

Table 5.2 Optimum Number of Reductions and Maximum Compression Ratio

arithmetic coder would make up much of the compression that was lost. The relative contribution of each (TP or DP) depends upon whether the image rendition is drawing/text or half-tone. In the next section the contributions of TP, DP and the coder are shown in pie chart form for each test image.

In drawings and text, including hand writing, TP accounted for nearly all the compression in the higher-resolution layers, the binary arithmetic coder provided most of the rest, and DP contributed negligibly. For half-tone images, the binary arithmetic coder accomplished most of the compression, DP added a significant amount, and the TP contribution was negligible.

5.2 Detailed Discussion

Figures 5.1 through 5.26 show detailed results, a pair of figures applying to each test image. The first figure in each pair shows the total number of compressed data bytes as a function of the number of image reductions.

The second figure in the pair shows one pie chart for each difference layer, giving the distribution of image bits over: bits not encoded because of TP and DP, encoded bits saved by the binary arithmetic coder, and bits actually transmitted. The whole pie represents the total number of pixels in the difference layer, plus the number of lines in the reference layer used while encoding the difference layer. The latter number is the number of TP exception bits, which are encoded in their own context, one per reference image line. Thus, the whole pie represents the total number of bits processed for one image layer.

Showing the contribution of Adaptive Templates (AT) in the pie charts was not feasible, because AT is an integral part of the binary arithmetic coding process. Document JBIG N139-R2 (reference cited) states that AC (AT) can sometimes halve the byte count in images in which it is most advantageous. A spot test on layer R1 of Image 4c showed a 9.7 percent reduction, and on layer R1 of Image 4d a 21.8 percent reduction. These two image layers had already been determined, during earlier simulations, to experience a floating pixel change early in the image scan. The comparisons were obtained by encoding the two layers as base layers with and without AT. The base layer mode was selected so that prediction would not contribute to the compression.

Figures 5.1 and 5.2 give the simulation results for Image 1, a text image.

The optimum number of reductions was 1, and the byte count was 13 percent below that for no reduction. TP accounted for most of the data compression, the binary arithmetic coder provided most of the rest, and DP added a small amount.

Figures 5.3 and 5.4 apply to Image 2, which contains text and drawings. Again, the optimum number of reductions was 1, with the byte count 8 percent below that for no reduction. TP accounted for almost all of the compression.

Figures 5.5 and 5.6 present the results for Image 3, Japanese writing. The compression for this image was an order of magnitude worse than for the first two images. The optimum number of reductions was 2, but the difference between 2 and 1 was less than one percent. The minimum byte count was 10 percent below that for no reduction. TP accounted for most of the compression in layers R0 and R1, with the binary arithmetic coder contributing most of the rest, and DP becoming significant in the lower resolution layers.

Images 4a, 4b, 4c and 4d are half-tone renditions of the same subject (sails), but were produced by different methods. Figures 5.7 and 5.8 provide the data for Image 4a, where the dither is 8x8. The optimum number of reductions was 1, with a 10 percent reduction in byte count with respect to no reduction. TP was minimal in all layers, DP contributed significantly, but the binary arithmetic coder contributed by far the most to the data compression.

Results for error diffused half-tone Image 4b are shown in Figures 5.9 and 5.10. The optimum number of reductions was 0 (i.e. base layer algorithm), with one reduction requiring 12 percent more bytes than none. TP was practically non-existent in all layers, DP contributed some compression, but, as in Image 4a, the binary arithmetic coder produced most of the compression.

The results for Image 4c, with 4x4 dither, are depicted in Figures 5.11 and 5.12. The optimum number of reductions was 1, and the improvement with respect to none was 19 percent. In layer R0, the binary arithmetic coder provided most of the compression, DP contributed substantially, and TP was negligible. DP decreased and TP increased in lower resolution layers, but the binary arithmetic coder contributed more compression than TP and DP combined in all layers.

Figures 5.13 and 5.14 show the results for Image 4d, produced by a 3x3 dither. The optimum number of reductions was none, with one reduction producing a total byte count 42 percent greater than none, a very significant difference. TP was negligible in all layers, DP contributed about the same amount in each layer, and the binary arithmetic coder, as usual for half-tone images, contributed most of the data compression.

The results for Image 5, hand writing, are given in Figures 5.15 and 5.16. The optimum number of reductions was 2, with 19 percent improvement with respect to none, and 4 percent with respect to one reduction. TP contributed the lion's share of the compression in all layers, the binary arithmetic coder providing most of the rest, and DP adding a small amount.

Figures 5.17 and 5.18 show the results for Image 6, a collage of German text, white on black and black on white, and small half-tone pictures. The optimum number of reductions was 1, with a total byte count 15 percent below that for none. In layer R0, TP and DP combined contributed roughly as much to the compression as the binary arithmetic coder, TP much more than DP. For lower-resolution layers, the coder contributed most of the compression. With decreasing resolution, the importance of TP decreased sharply; that of DP varied slightly.

For Image 7, a scanned line drawing, Figure 5.19 shows that the optimum number of reductions was 2. The byte total was 7 percent less thar that for no reduction, and 1 percent less than for one reduction. The pie charts of Figure 5.20 show that TP dominated the compression, the coder provided most of the rest, and DP contributed slightly.

Figures 5.21 and 5.22 apply to Image 8, a generated line drawing. It is interesting to note that the computer generated drawings (Images 8 and 10) gave roughly 2.5 times the compression ratio as the scanned drawing of Image 7. For Image 8, the optimum number of reductions was an exact tie between 1 and 2. The minimum byte count was 10 percent below that for no reduction. Again, TP contributed to most of the compression, with the coder providing most of the rest, and DP adding a slight amount.

Image 9, another error diffused image, gave results shown in Figures 5.23 and 5.24. As in the other error diffused image and the 3x3 dither, the optimum number of reductions was 0, and one reduction required 15 percent more total compressed data bytes than did none. TP was virtually nonexistent, DP contributed roughly the same amount of compression in all layers, and the coder provided the most in all layers except R4.

Finally, Figures 5.25 and 5.26 show the results for Image 10, another computer generated "line drawing" (actually a pie and a bar chart with small circles, grids and hatches used for fill). The optimum number of reductions was 2, with a 13 percent reduction in total byte count with respect to no reduction, and 3 percent with respect to one reduction. TP provided nearly all of the compression in the higher resolution layers, the coder produced most of the rest, with DP contributing slightly.

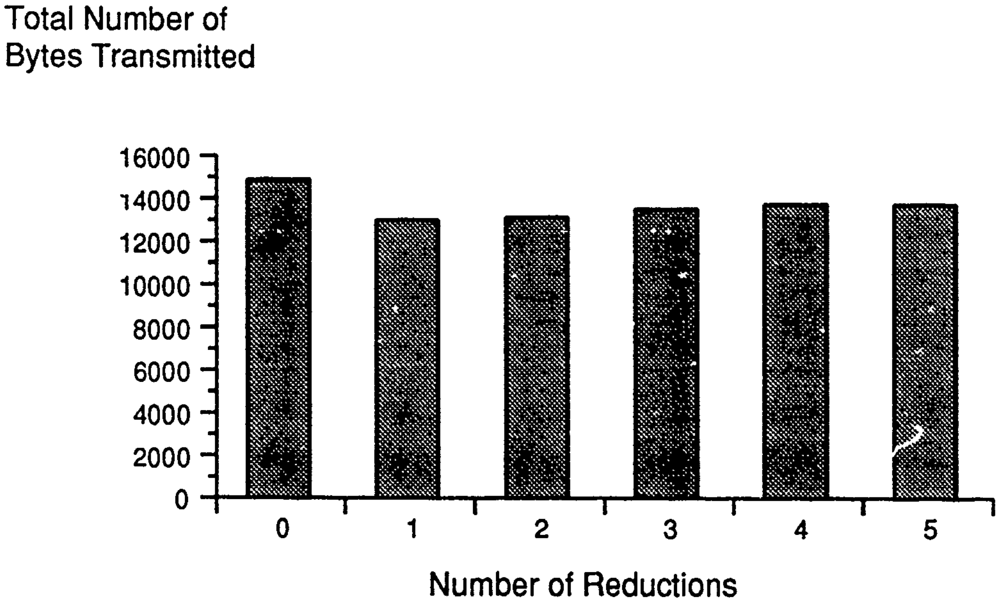


Figure 5.1 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 1

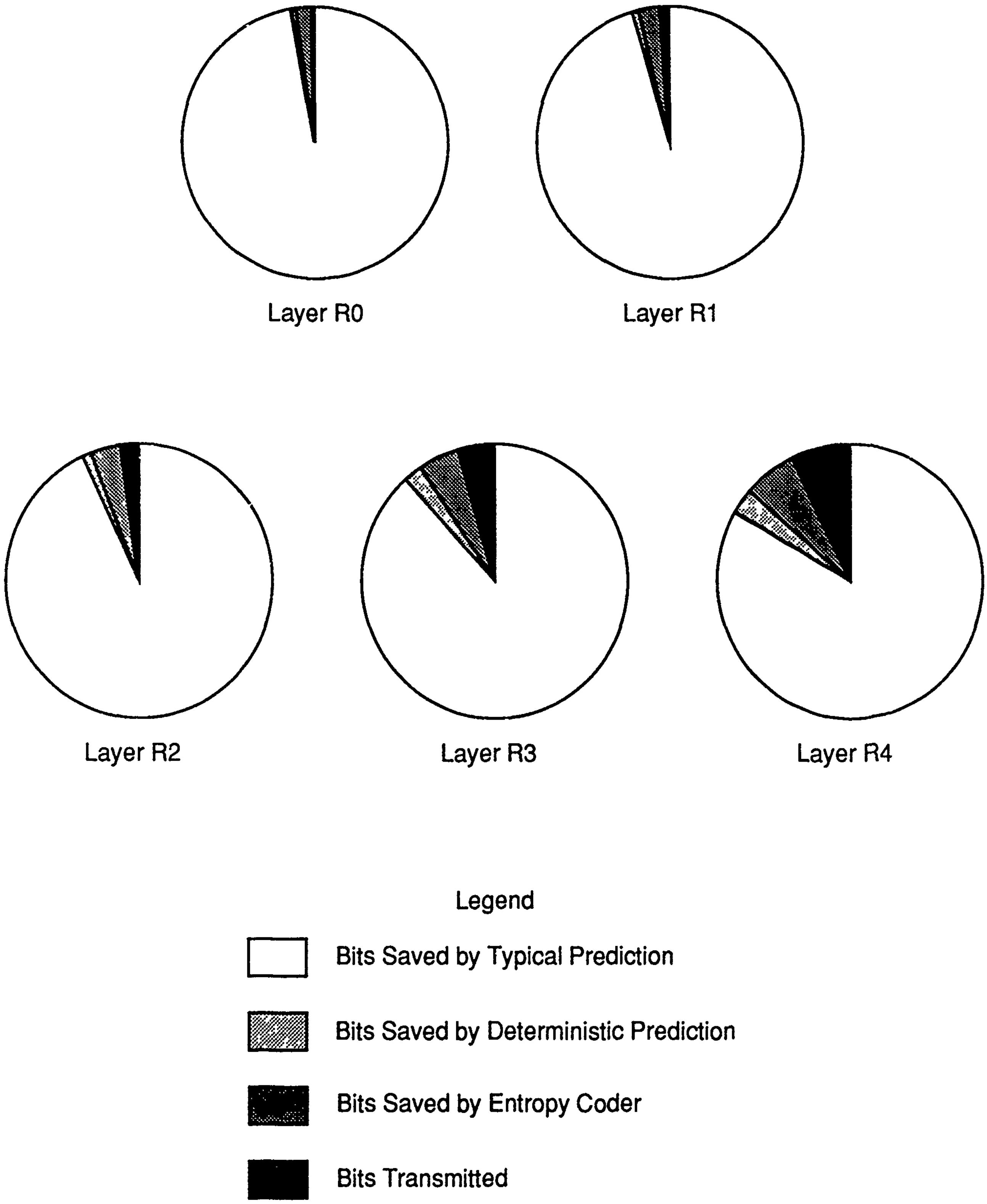


Figure 5.2 DIstribution of Data Bits for Test Image 1 Difference Layers

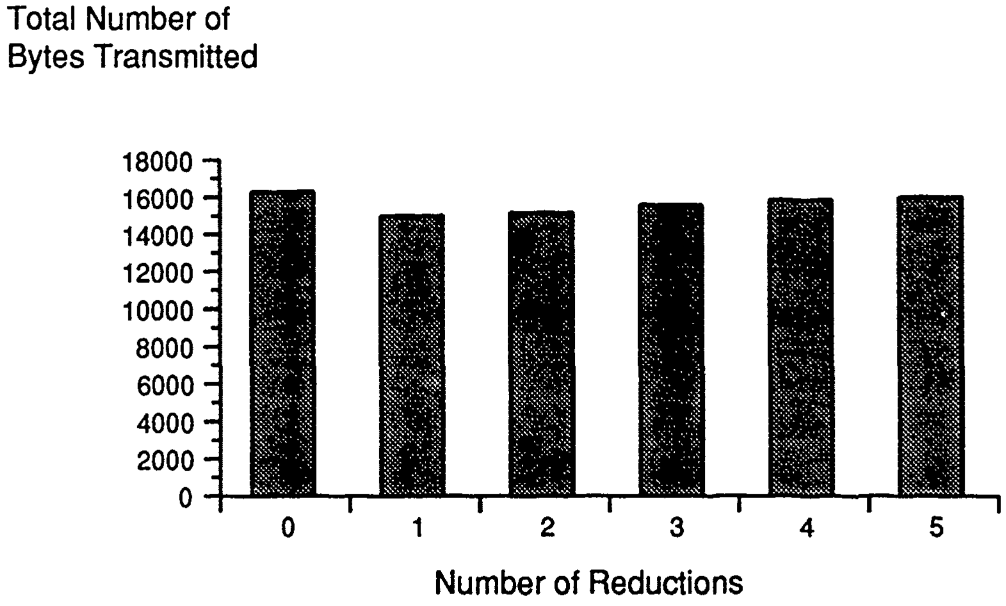


Figure 5.3 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 2

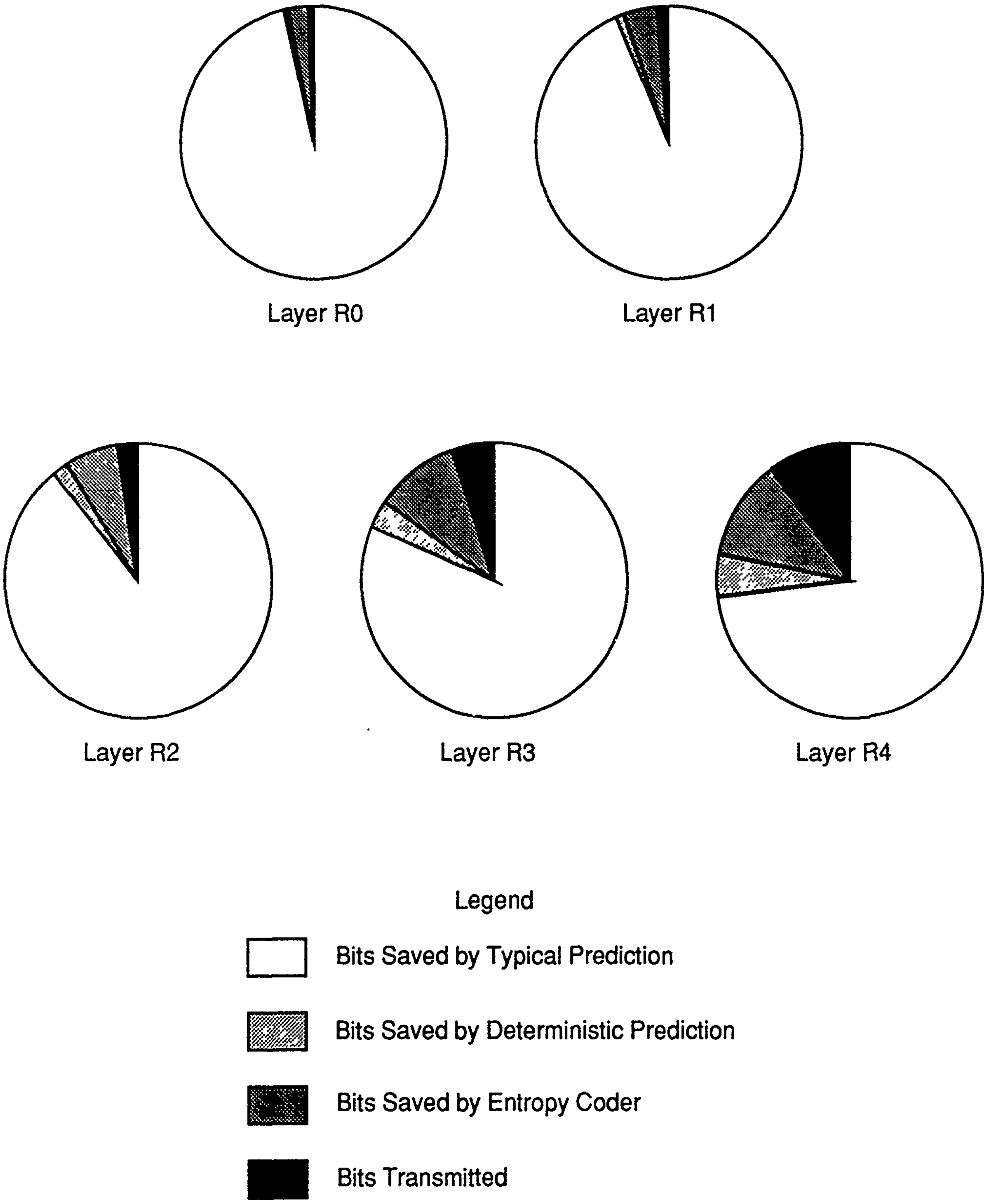


Figure 5.4 DIstribution of Data Bits for Test Image 2 Difference Layers

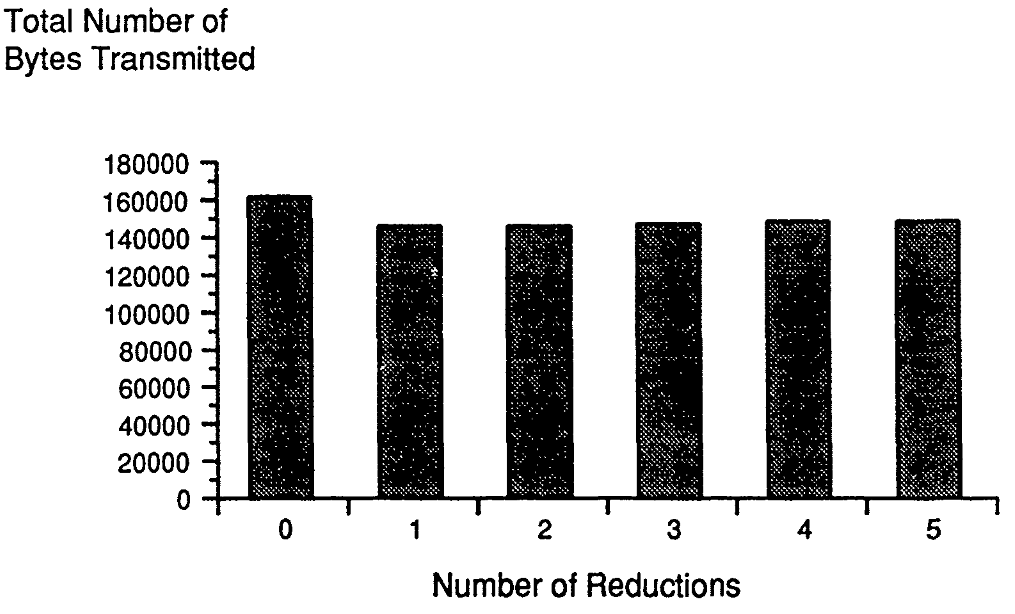


Figure 5.5 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 3

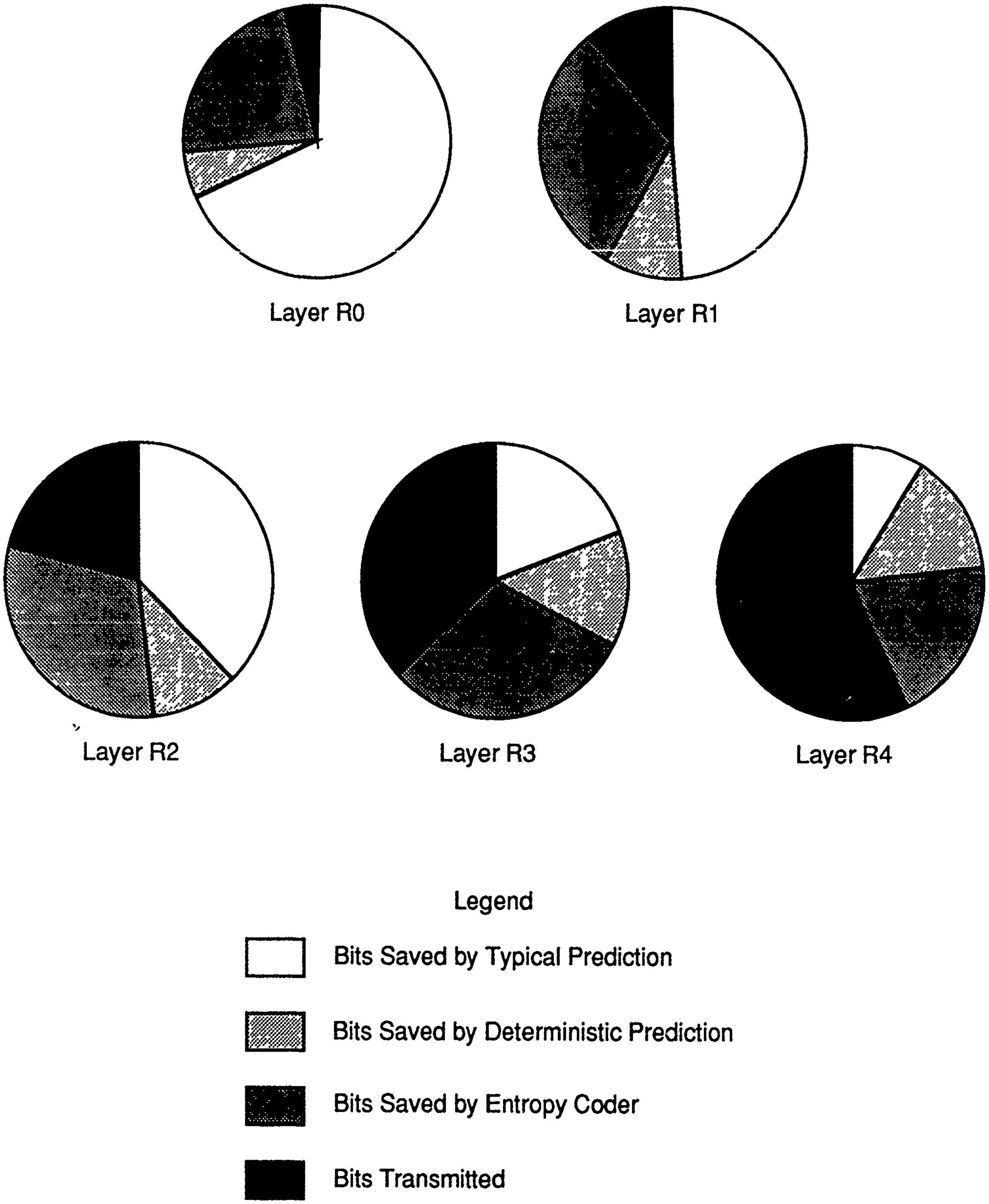


Figure 5.6 DIstribution of Data Bits for Test Image 3 Difference Layers

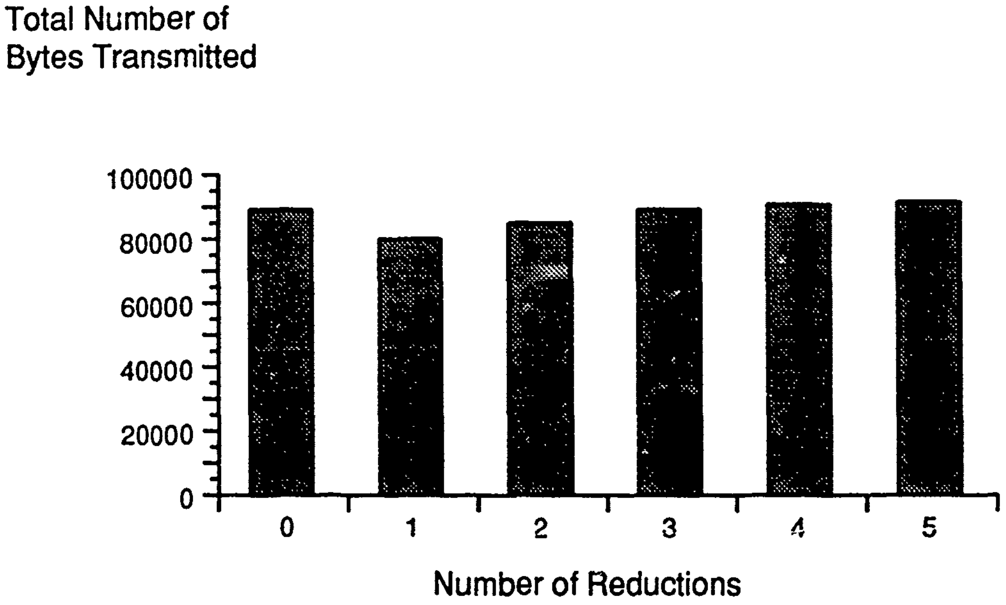


Figure 5.7 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 4a

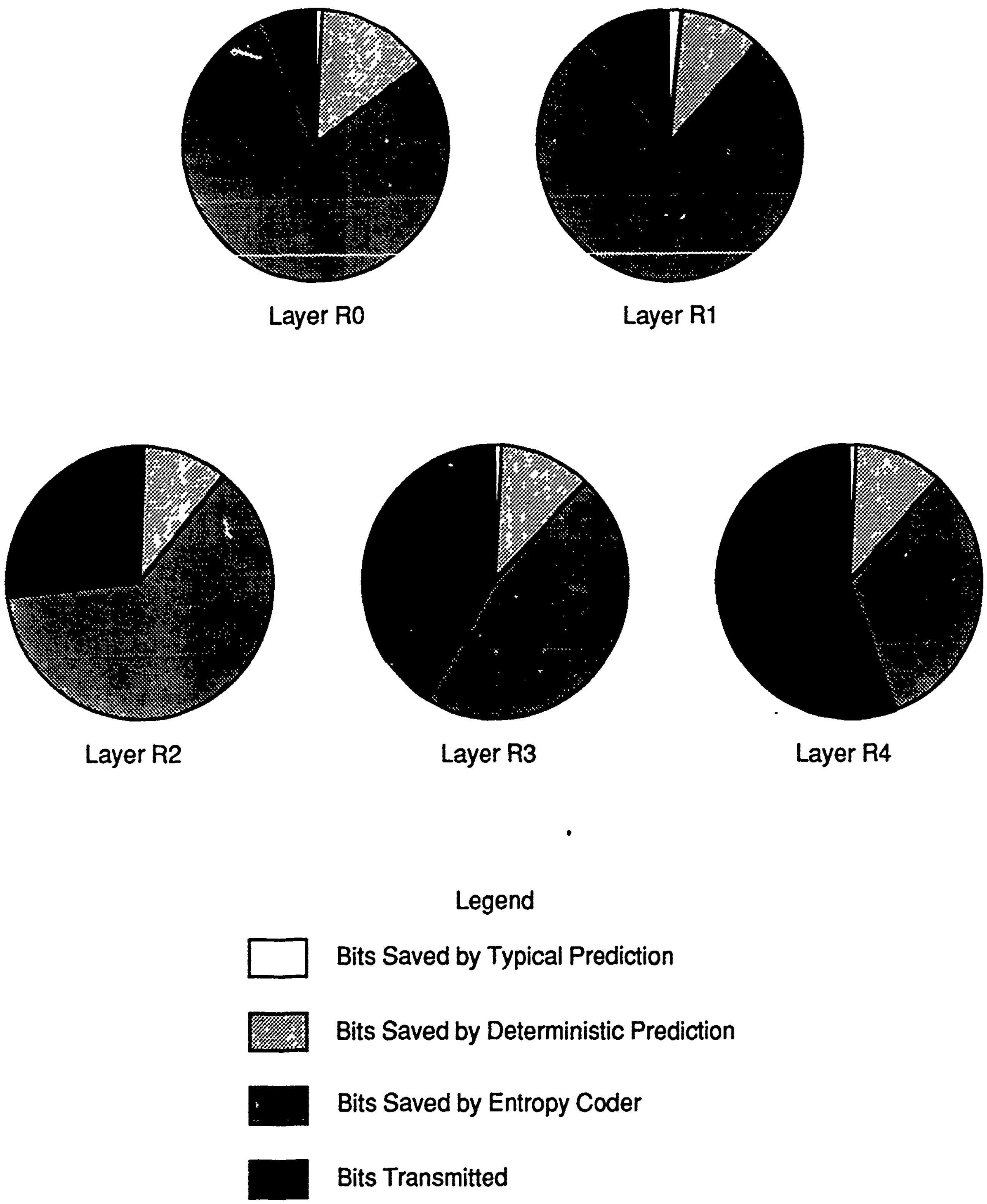


Figure 5.8 DIstribution of Data Bits for Test Image 4a Difference Layers

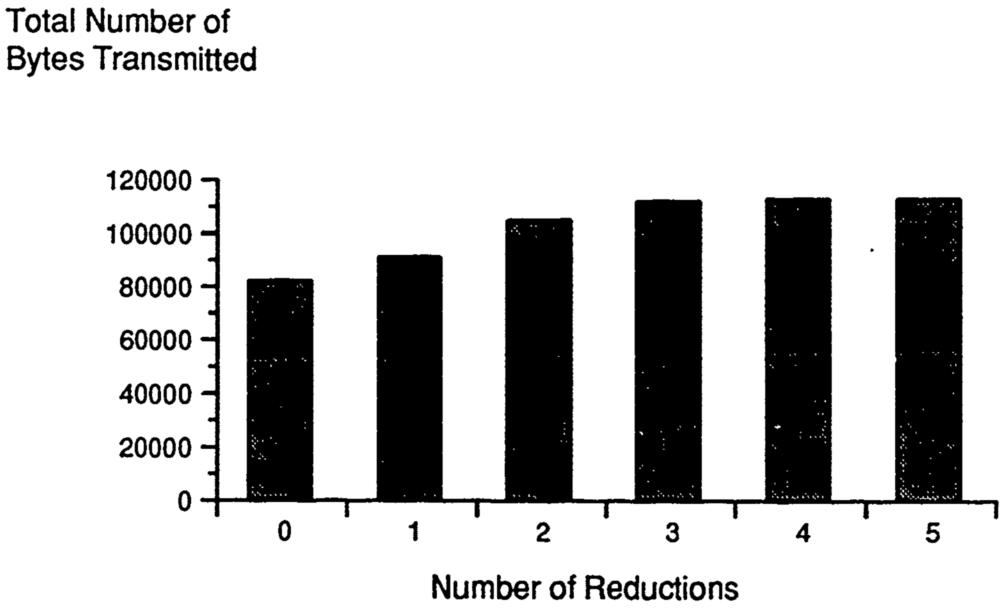


Figure 5.9 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 4b

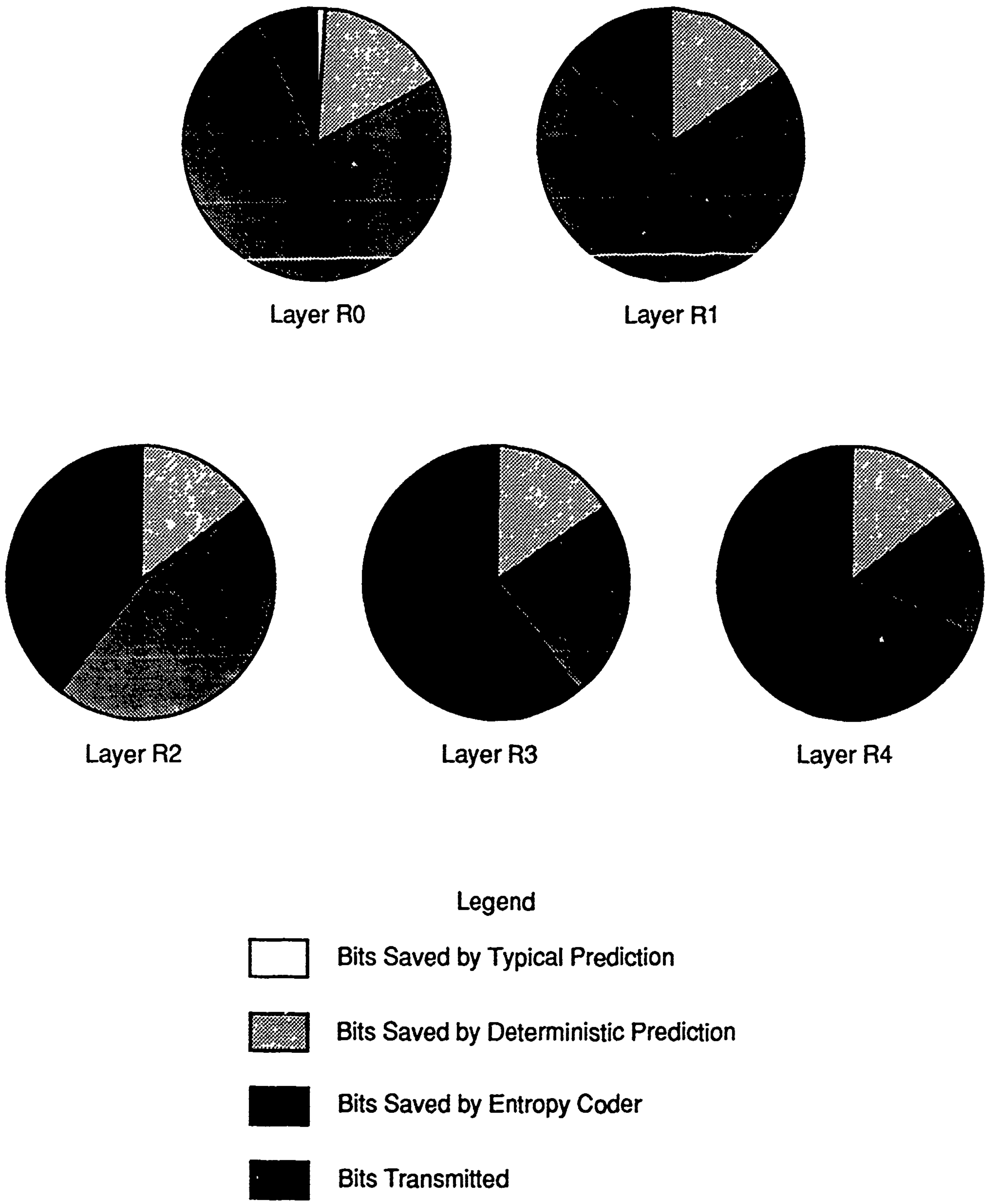


Figure 5.10 DIstribution of Data Bits for Test Image 4b Difference Layers

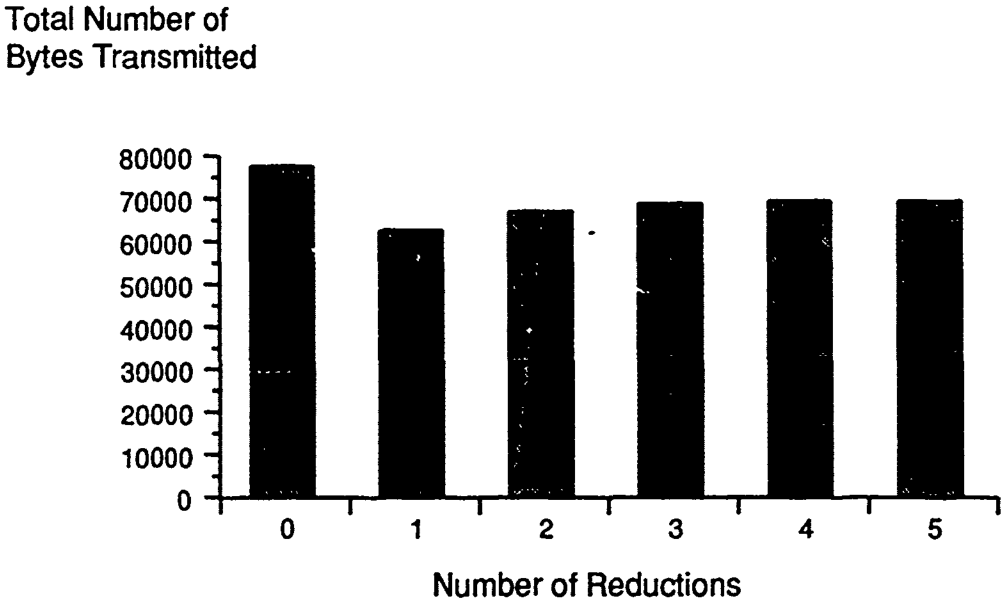


Figure 5.11 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 4c

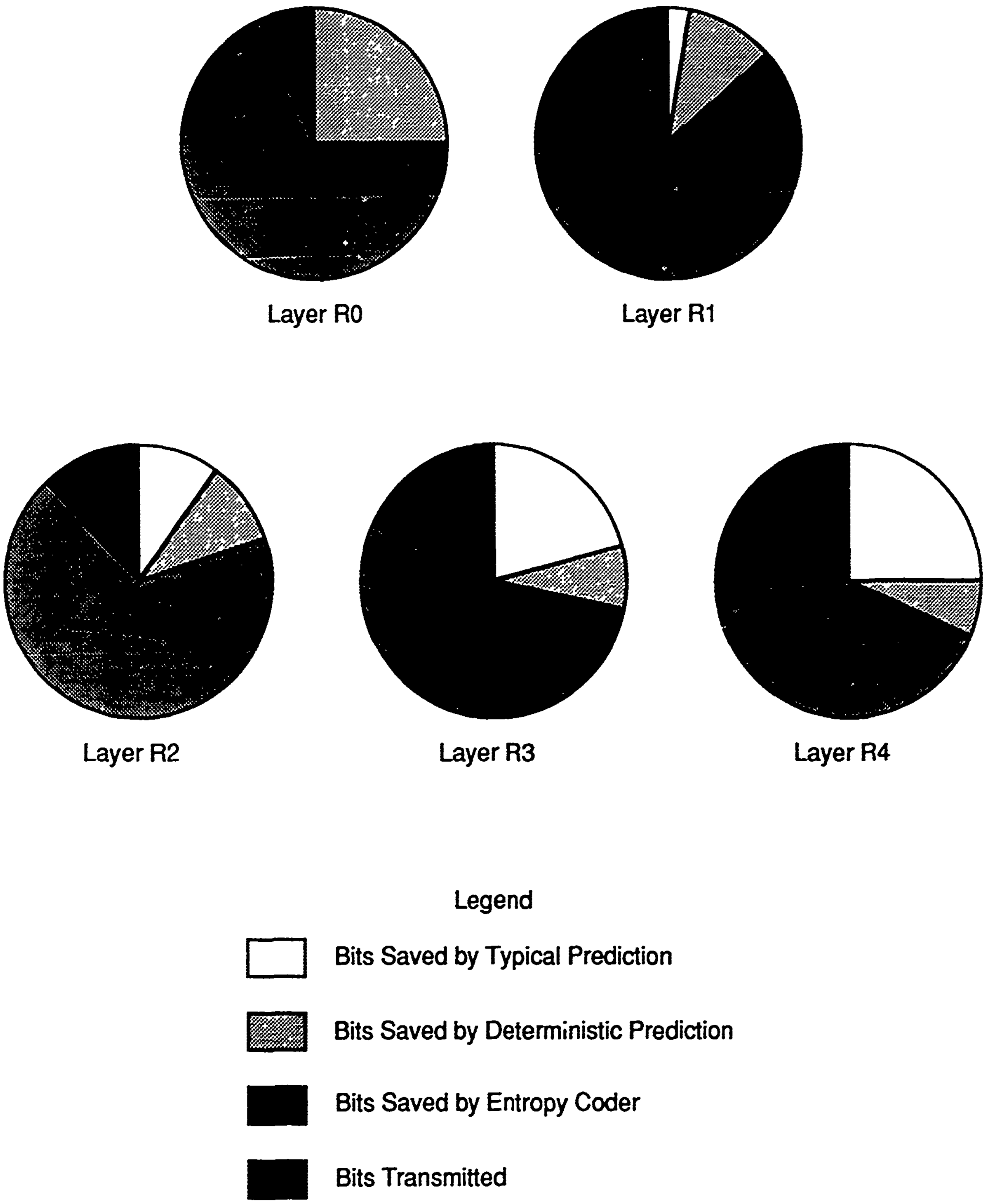


Figure 5.12 DIstribution of Data Bits for Test Image 4c Difference Layers

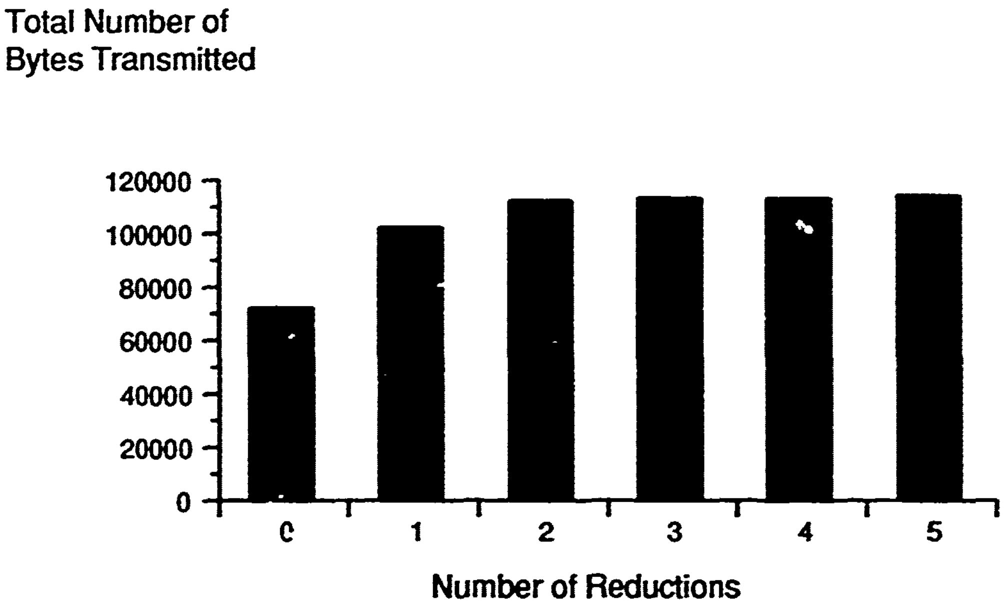


Figure 5.13 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 4d

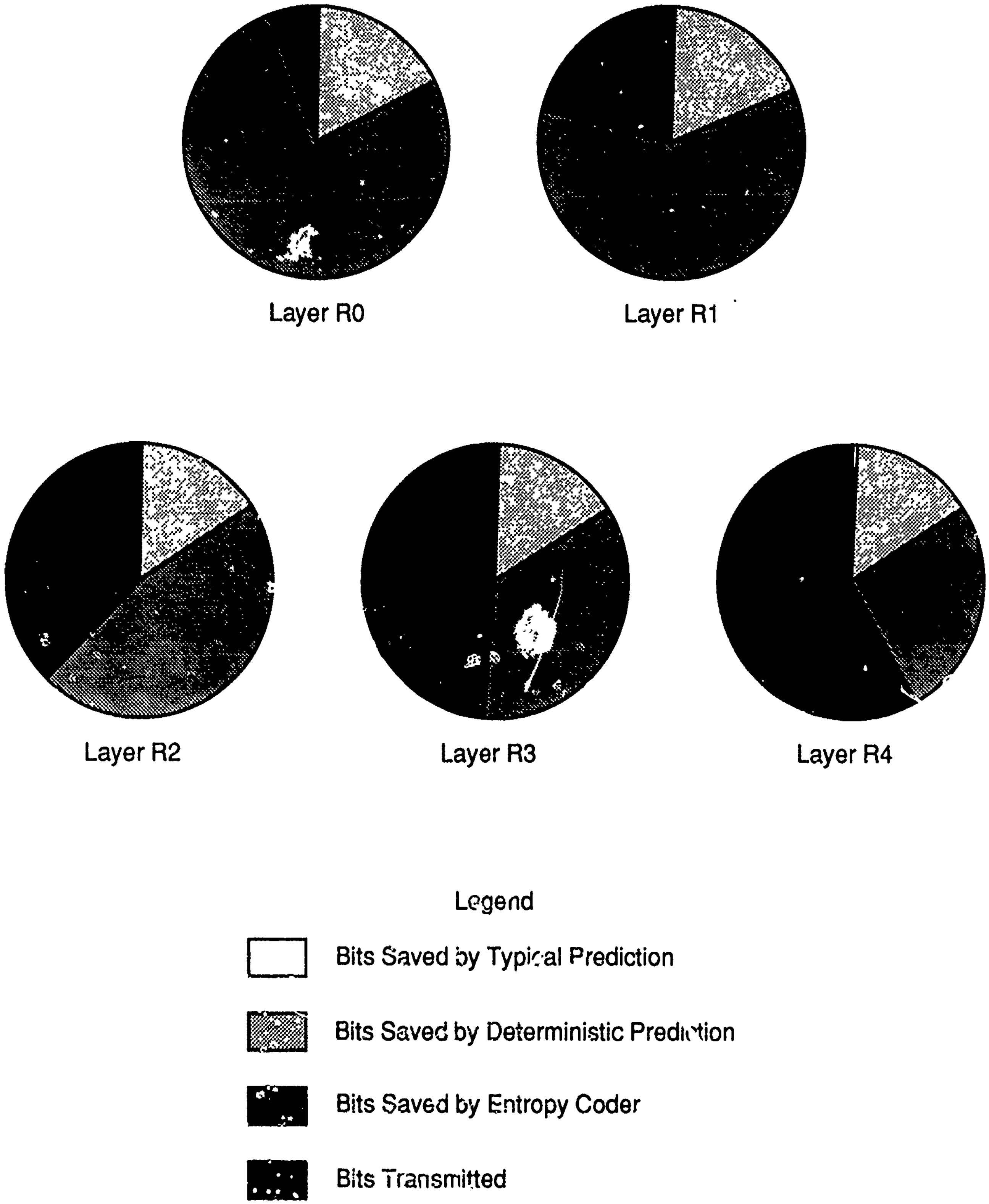


Figure 5.14 Distribution of Data Bits for Test Image 4d Difference Layers

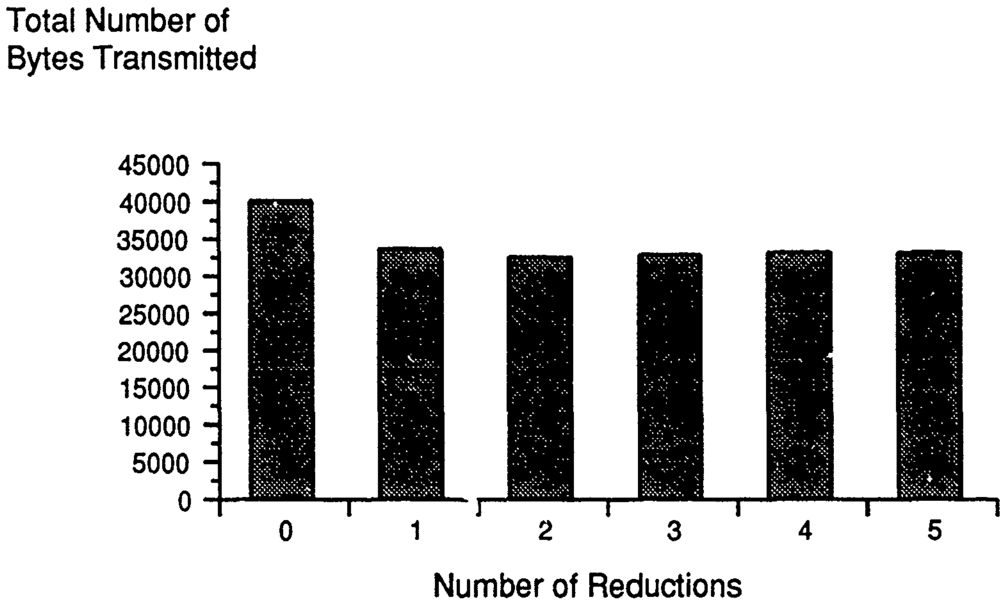


Figure 5.15 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 5

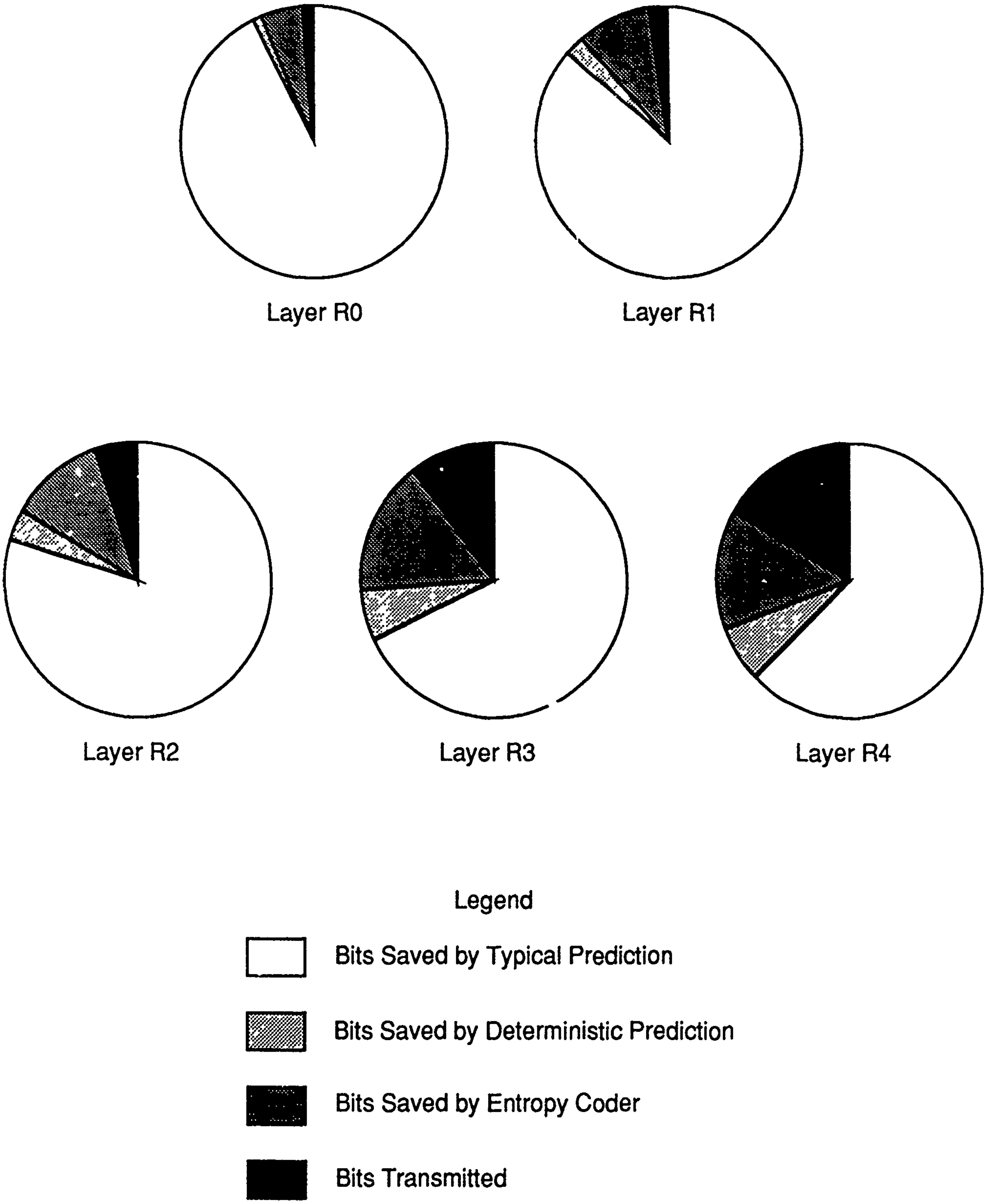


Figure 5.16 DIstribution of Data Bits for Test Image 5 Difference Layers

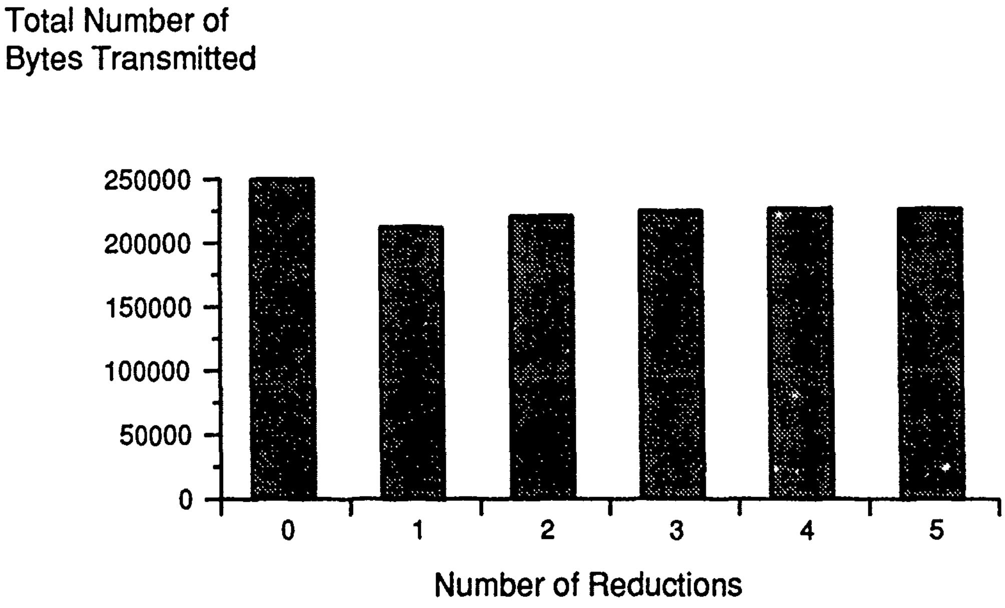


Figure 5.17 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 6

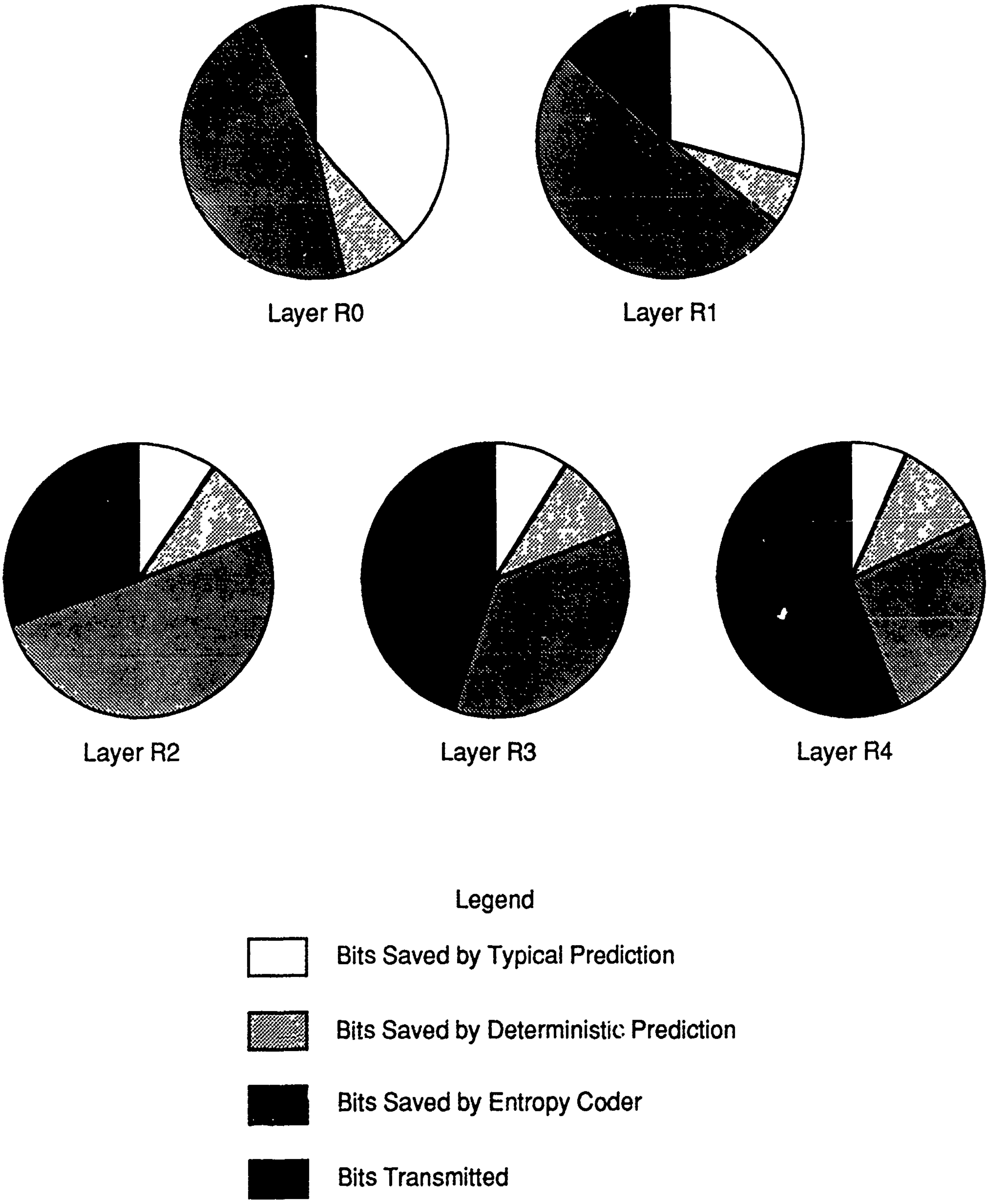


Figure 5.18 DIstribution of Data Bits for Test Image 6 Differance Layers

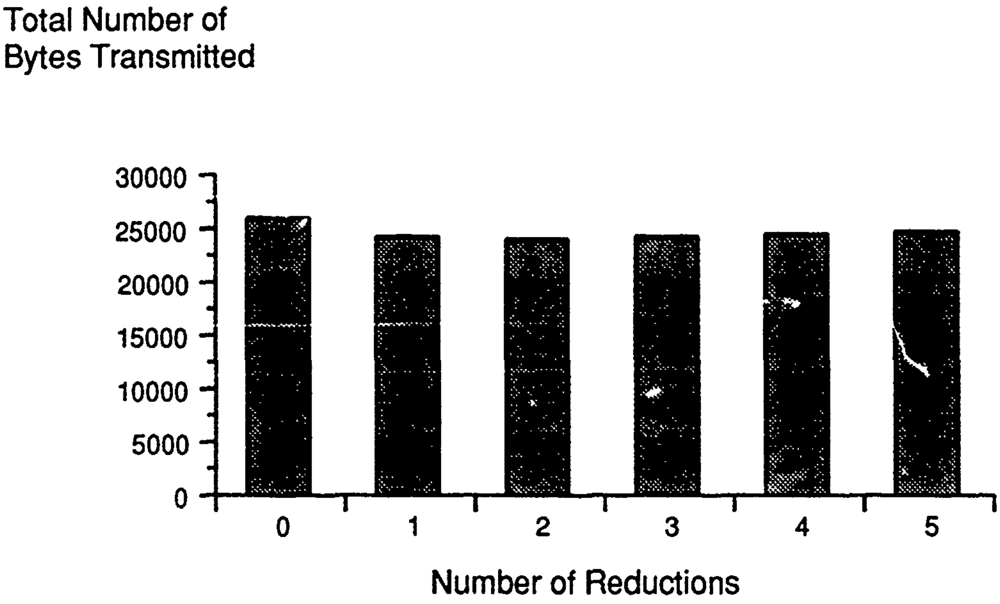


Figure 5.19 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 7

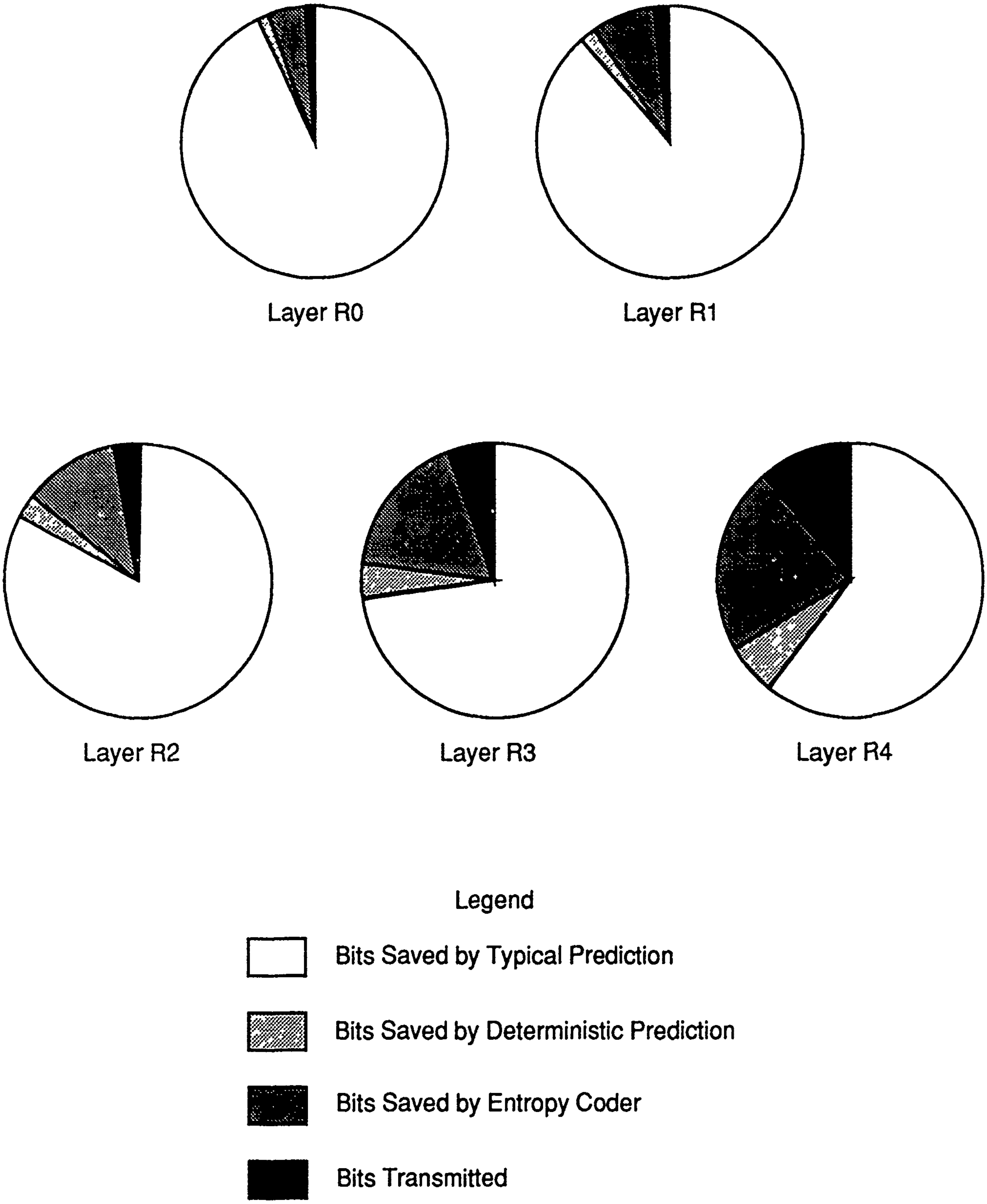


Figure 5.20 DIstribution of Data Bits for Test Image 7 Difference Layers

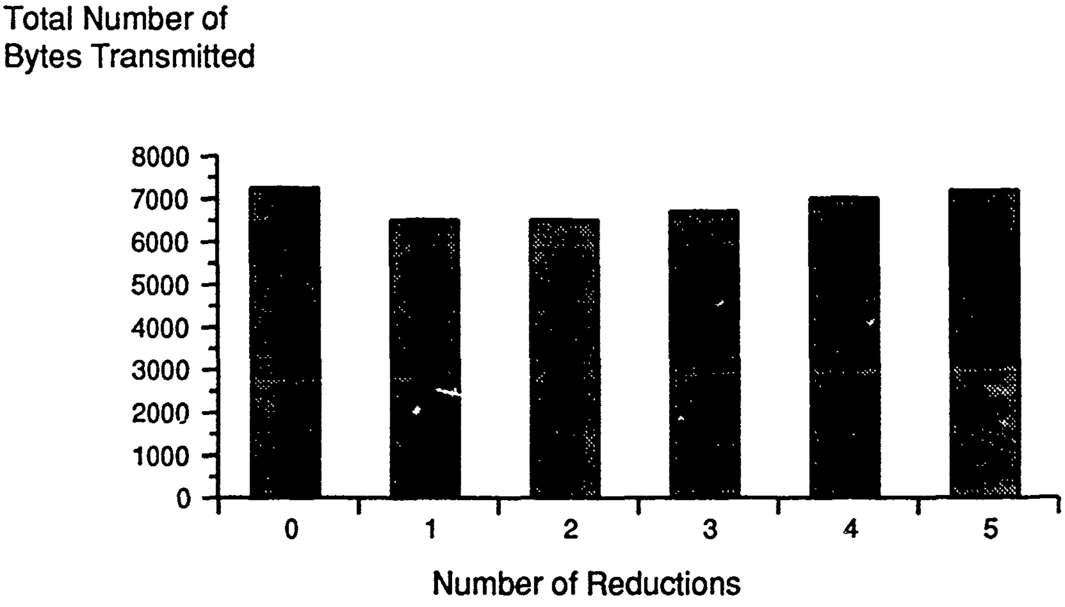


Figure 5.21 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 8

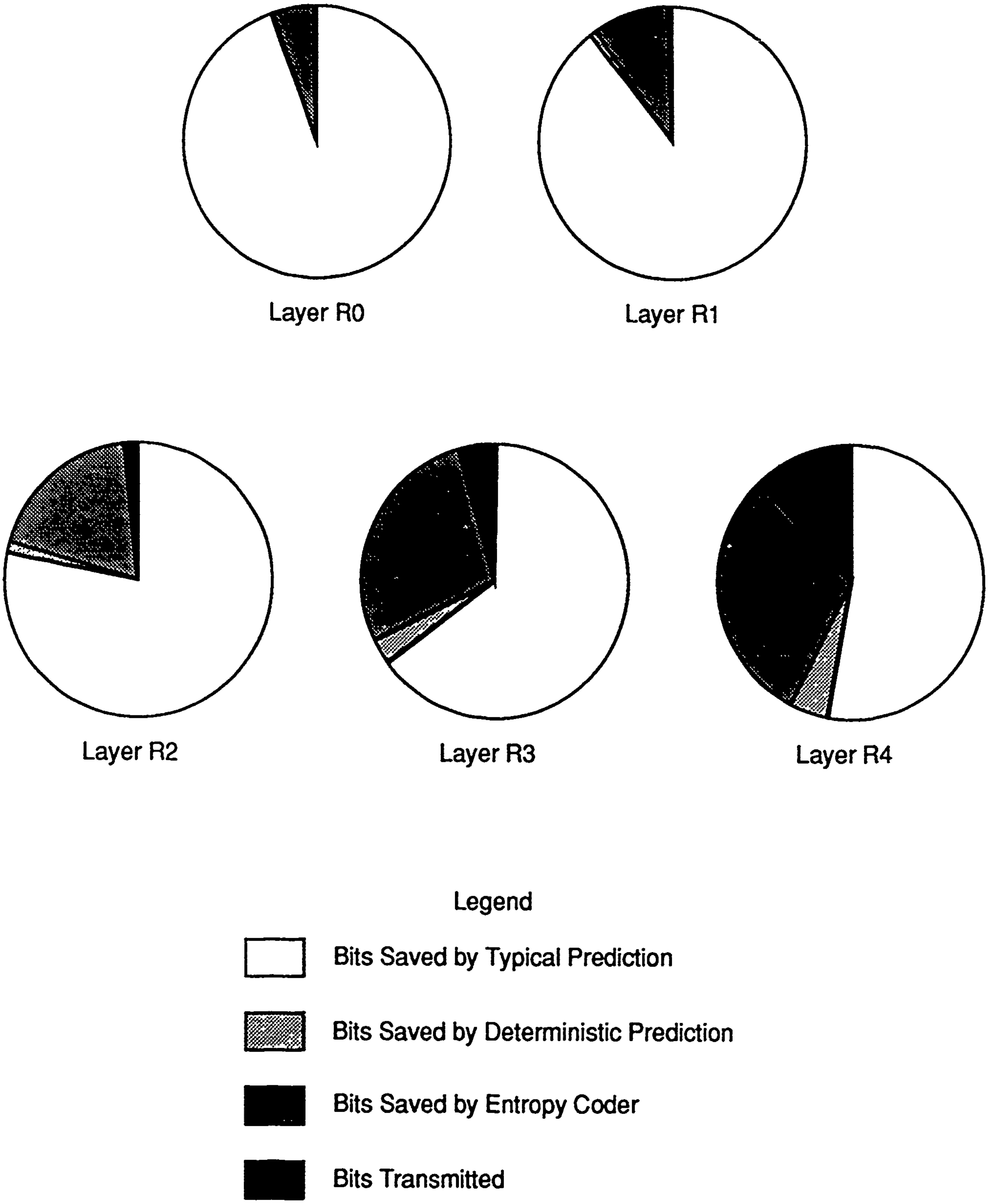


Figure 5.22 DIstribution of Data Bits for Test Image 8 Difference Layers

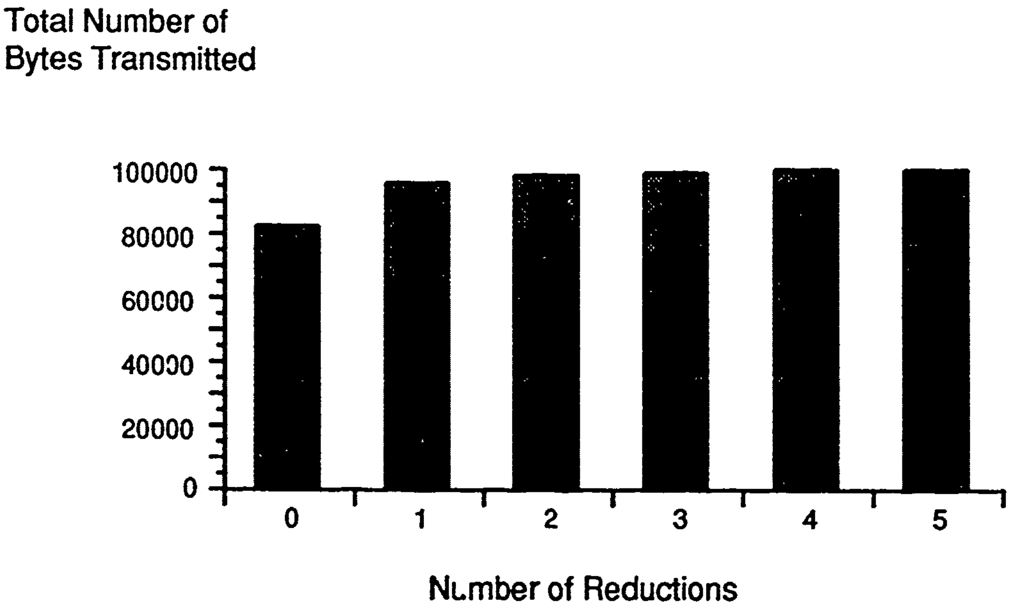


Figure 5.23 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 9

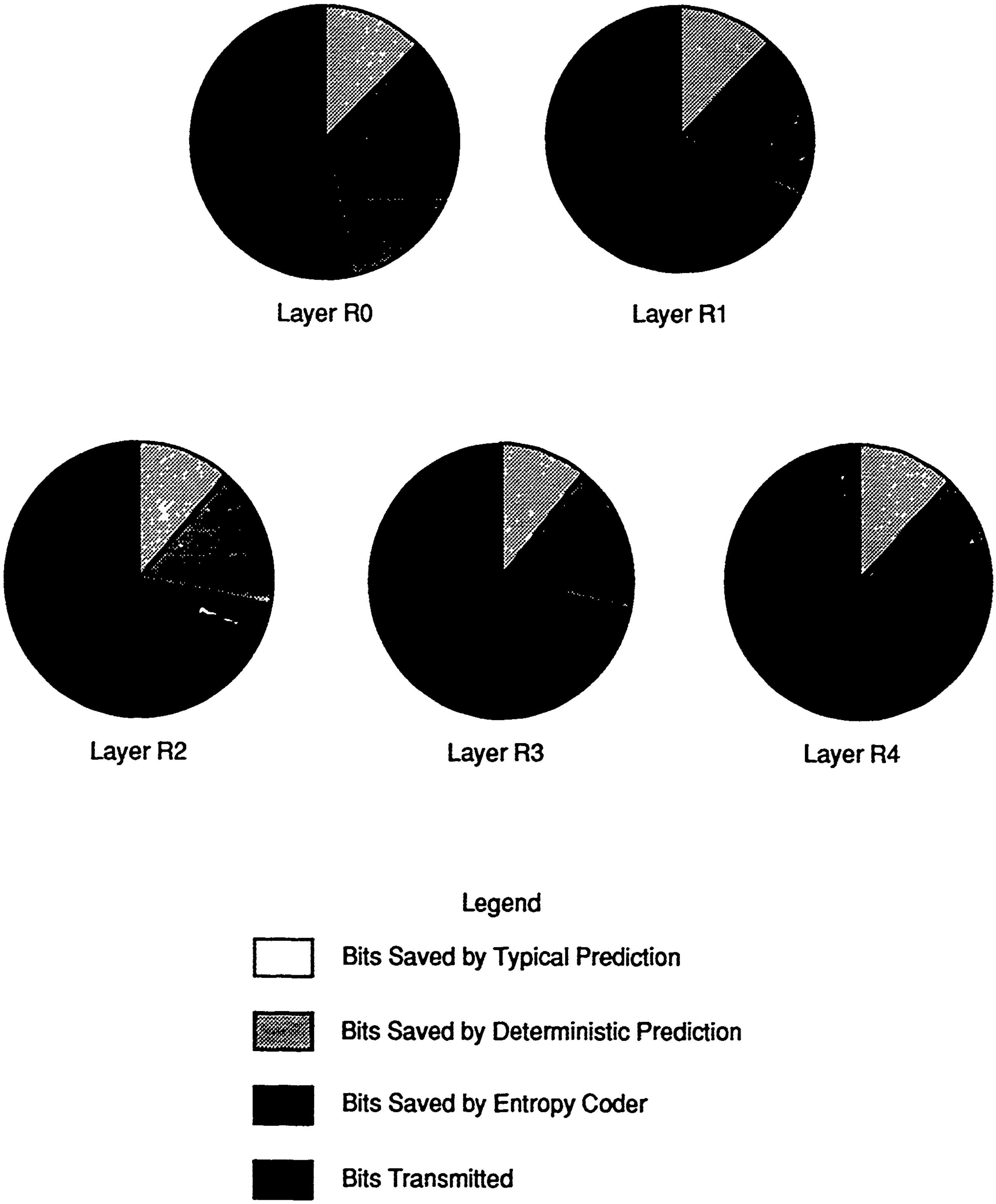


Figure 5.24 DIstribution of Data Bits for Test Image 9 Difference Layers

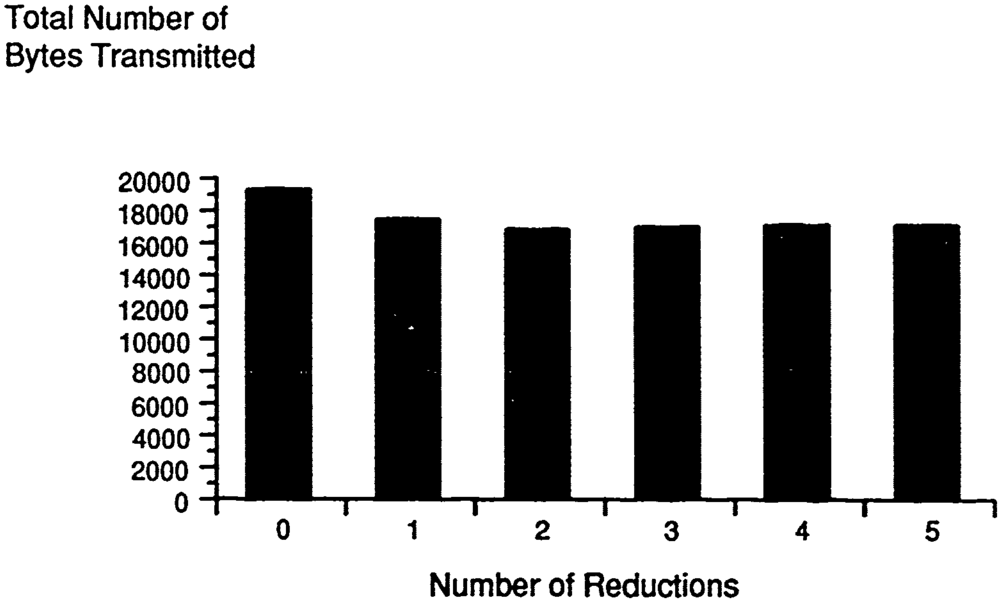


Figure 5.25 Total Number of Bytes Transmitted vs. Number of Reductions for Test Image 10

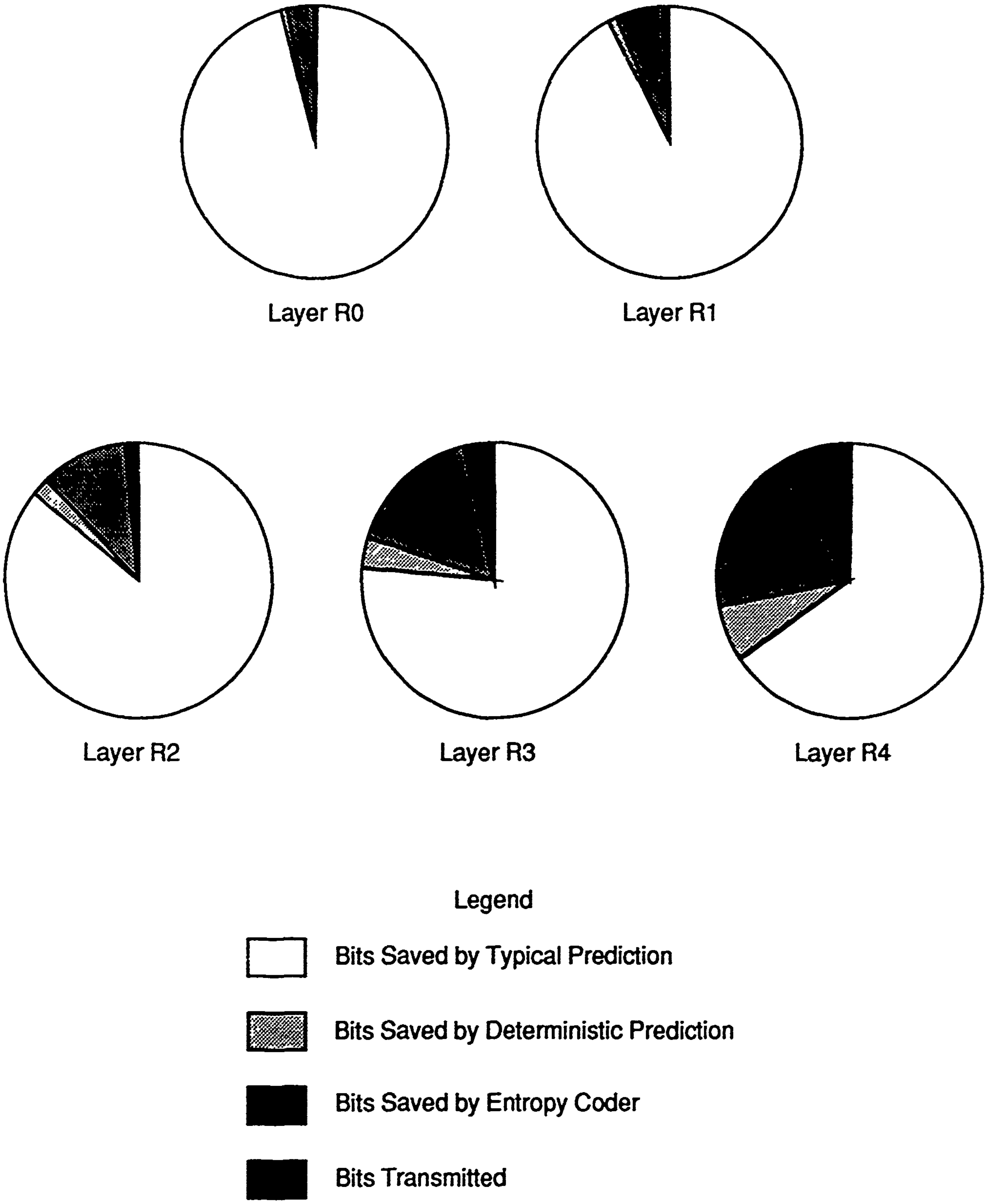


Figure 5.26 DIstribution of Data Bits for Test Image 10 Difference Layers

6.0 CONCLUSIONS

The information gleaned by simulating the JBIG Base System algorithm in the progressive mode and by performing partial progressions with fewer than 5 image reductions, including none, leads to the following conclusions.

Typical prediction (TP), deterministic prediction (DP) and adaptive context templates (AT) all contribute to compression, some more than others, depending upon resolution and whether or not the image is half-tone. TP contributes the most compression in line drawings and text images, including hand writing, but is of negligible importance in half-tone images. DP and AT both contribute significantly in half-tone images, but negligibly in drawings and text. Note that these compression results are based on the application of all parts (TP, DP, AT) of the algorithm. This does not mean that similar overall compression could not be achieved by omitting part (i.e. TP) and making up the compression with another part (i.e. the arithmetic coder). However, to achieve the highest compression in a progressive system designed to be independent of image type, TP and DP should both be implemented in difference layer processing, and AT should be implemented in difference and base layer processing, at least if the present JBIG templates are employed.[8]

The binary arithmetic coder greatly reduces the number of transmitted pixels with respect to those which must be encoded (are neither TP nor DP) regardless of image type; hence, it is concluded that the JBIG context templates are well chosen. This last conclusion was further substantiated by observing the coder LPS probability estimates, averaged over the contexts, at various points during and at the end of each coding task. The MPS/LPS probability skew rapidly becomes high and reaches roughly a steady state, fluctuating slightly with varying local statistics as coding progresses. It was also observed that the skew, and hence the compression (of the encoded data, not TP or DP pixels), is highest in the highest resolution images, and decreases as the resolution becomes lower. It is conjectured that this is because PRES, which seeks to preserve detail during image reduction, removes a great deal of redundancy; hence, the context templates become poorer predictors as resolution

[8] JBIG Document No. 190, "Using the JBIG Starting Layer Algorithm for All Resolutions," reference cited, suggests that AT can be eliminated if the AT lag pixels are always included in the context template.

decreases.

Whether the starting layer algorithm can compete with a progression consisting of one or more reductions is still an open question. For 400 dots per inch resolution, the starting layer algorithm (with the current JBIG base layer template) gave worse compression in most cases than a progression involving at least one image reduction. Over all 13 test images, at least one reduction gave an average percentage decrease in total byte count of 4.2 with respect to no reduction. When at least one reduction was better, typical improvements were 10 to 20 percent, but in the three cases in which no reduction was best, the degradations incurred by performing one reduction were 12, 15 and 42 percent.

The starting layer algorithm was better for most of the images at resolutions of 200, and always better at resolutions of less than 200 dots per inch. This follows directly from the fact that the total byte count increased in most cases when the number of reductions exceeded one, and always when the number exceeded two.

The computation burden of the JBIG Base System is intense. PRES must examine 9 high- and 3 low-resolution pixels for each low-resolution pixel it produces. Proving that a TP cluster exists and is not an exception requires examining 9 low- and 4 high-resolution pixels. Testing whether one pixel is DP requires assembling into a state an average of 10 other pixels, the actual number depending upon phase. Determining contexts requires assembling 7 pixels for a base layer, 10 for a difference layer, per pixel to be encoded. AT requires maintaining correlation counts for 7 AT pixels per target pixel.

If the starting layer algorithm can be made to perform as well, or almost as well, as at least one reduction, then PRES, TP and DP can all be eliminated. (PRES can be used by itself to perform image reduction when needed for purposes other than transmitting an image at a given resolution.) Use of context templates that always include the present AT pixels would obviate the need for AT. It is therefore recommended that the starting layer algorithm be vigorously pursued, at least for simple facsimile transmission.

For data base storage, browsing and retrieval, with various resolutions from icons to full-scale images, the full JBIG algorithm is appropriate. The data base system could employ the full algorithm to decompress and expand an image to any desired resolution, and then use the starting layer algorithm to recompress and transmit the expanded image to a user site. This would allow the many data base users to have relatively inexpensive terminals (including current facsimile machines), with the data base system having the "intelligence" required to store and retrieve images of varying resolution. Similar functionality could provide backward compatibility to the large installed base of Group 3 facsimile terminals. If the JBIG functionality would be added to terminals that had softcopy interactive capability, such as Group 4, then users of such terminals would realize the full benefits of the progressive algorithm.

REFERENCES

1. JBIG Document N204-RO, "Draft JBIG Technical Specification"

2. JBIG Document N198, "PRES Specification." March, 1990

3. JBIG Document N140, "DP2 Report: Z-Scan Versus Row Scan," September 4, 1989.

4. JBIG Document N177 Rev. 1, "Resolution of 6th JBIG Meeting" (October 16 - 24, 1989 in Japan, no document date given)

5. JBIG Document N31, "Progressive Coding Scheme Using Block Reduction (PCSB)," September 15, 1988

6. JBIG Document N32, "PED: Progressive Edge Decomposition of Facsimile Images," September 15, 1988

7. JBIG Document N75, "Progressive Coding Method for Bi-Level Images," January, 1989

8. JBIG Document N130, "Typical Prediction, A Preprocessor for Progressive Coding of Bilevel Images," June 26, 1989

9. JBIG Document N131, "A Simple Way for Transmitting the Exceptions of the TP Preprocessor," June 29, 1989

10. JBIG Document N199, "DP (Deterministic Prediction) for PRES," March, 1990, also JBIG Document N141, "DP Pattern List of PRES," October, 1989

11. ibid

12. JBIG Document N174, "Template Committee Report," no date given

13. JBIG Document N139-R2, "More Information on Adaptive Contexts," December 5, 1989

14. JBIG Document N208-R0, "JBIG Technical Specifications for Adaptive Templates," no date given

15. ANSI/X3L2.8 Doc. No. X3L2/89-107, "QM: A Common Adaptive Binary Arithmetic Coder for JBIG and JPEG," September 12, 1989

16. JBIG Document N158-R0, "Basis for a 'QM' Coder," October 4, 1989

17. JBIG Document N213-R0, "Adding the Minimax Fast Attack to the JPEG Arithmetic Coder," March 12, 1990

18. See, for example, W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, Jr. and R. B. Arps, "An Overview of the Basic Principles of the Q-Coder Binary Arithmetic Coder," IBM J. Res. Develop., 32, 717-726, 1988, or Mitchell and Pennebaker, "Optimal Hardware and Software Arithmetic Coding Procedures for the Q-Coder," IBM J. Res. Develop., 32, 727-736, 1988.

19. Document X3L2.8/90 - 004, "JPEG Draft Technical Specification (Revision 5)," January 15, 1990

20. JPEG-8-R5.2, "Draft (Revision 5.2) of the JPEG Algorithm," May 10, 1990

21. JBIG Document No. 190, Rev. 1, "Using the JBIG Starting Layer Algorithm for All Resolutions," March 12, 1990

22. JBIG Document N219, "'Pure' Postprocessing for JBIG Arithmetic Coder," March 15, 1990

23. JBIG Document 105 Rev. 2, "Description of JBIG Evaluation Images for July Meeting [in Stockholm]," June 6, 1989

Appendix A

Principles of Binary Arithmetic Coding

Binary arithmetic coders are based on the principles presented here. The implementations and performances of specific coders are, however, widely varied.

Every binary arithmetic coder/decoder has a probability estimator. Assume, for simplicity, only one "context," that is, at any given time, there is only one probability estimate based on past binary symbols in the string being encoded or decoded. The extension to more than one context is accomplished by keeping separate statistics for each, as described in the body of this report.

Let MPS denote that binary symbol which is currently estimated to be the more probable; let LPS denote the other, less probable, binary symbol. Let P denote the current MPS probability estimate, and Q the current LPS probability estimate.

Let A denote a probability interval, initially containing all values greater than or equal to 0 and less than 1. This interval is denoted by [0,1), where [a,b) means that the interval includes a, but not b. Let C denote a code point, i.e., a number in the interval [0,1). The value of C is what is transmitted, i.e., a long binary fraction.

Interval A is divided into two sub-intervals, one representing the LPS probability, and the other the MPS probability. Let the LPS and MPS sub-intervals be L and M respectively. Without loss of generality, assume that L lies below M. A description similar to what follows would apply to the opposite case, in which the lower sub-interval represents the MPS, and the upper the LPS probabilities.

Assume, for the present discussion, a fictitious coder and decoder that can perform arithmetic with infinite precision, that is, express any value in [0,1) exactly.

In the encoder, A is initialized to [0,1), C to 0 and P and Q to 0.5. (Other values can be assigned to P and Q if there is a priori knowledge.) The initial value of the MPS (1 or 0) is assigned arbitrarily (unless there is a priori knowledge), since the two symbols are assumed, at the outset, to be equally probable.

When the encoder receives a binary symbol to be encoded, it does the following, based on whether the symbol is the MPS or the LPS:

```
MPS:
   Adds the LPS sub-interval to C,
   Sets $A = A * P$ (the asterisk denotes multiplication).
LPS:
   Leaves C alone,
   Sets $A = A * Q$.
Either case:
   Sets LPS sub-interval = $Q * A$,
   Sets MPS sub-interval = A - LPS sub-interval.
```

During the encoding/decoding process, the probability estimates are frequently updated. If what was the LPS becomes more probable than the MPS, the value of the MPS is reversed. Thus P and Q always denote the probabilities of the MPS and the LPS respectively, with Q less than or equal to P.

The decoder is similarly initialized, except that C contains the final value left by the encoder. The required precision of C will be presented momentarily.

The decoder decodes a binary value as follows:

```
If C is greater than or equal to the LPS sub-interval, then it:
    Decodes the MPS value,
    Subtracts the LPS sub-interval from C,
    Sets $A = A * P$.
else it:
    Decodes the LPS value,
    Leaves C alone,
    Sets $A = A * Q$.
either case:
    Sets LPS sub-interval = $Q * A$,
    Sets MPS sub-interval = A - LPS sub-interval.
```

A study of the encoding process reveals that, at any given time, A represents the probability that a particular string of binary decisions was encoded prior to that time, M is the probability that this string was encoded and that the next symbol will be the MPS, and L is the probability that this string occurred and that the LPS will be encoded next. As each symbol is encoded, the new A is in effect laid beside the old sub-interval associated with that symbol. As an MPS is encoded, the new A is laid beside the M sub-interval of the old A; as an LPS is encoded, the new A is laid beside the L sub-interval. C points to the bottom of the new A interval. Furthermore, at any given time, the interval [C,C+A) is included in the sub-intervals of all the symbols encoded prior to that time. Therefore, the decoding process can begin with any value C' in the final interval [C,C+A) left by the encoder without affecting the ability of the decoder to reconstruct the original string of binary decisions. The minimum number of bits required in the compressed data stream is therefore the minimum number required to express a value of C' anywhere in the final interval [C,C+A).

Figure A.1 shows graphically, approximately to scale, the encoding of 4 symbols: MPS, MPS, LPS, MPS (e.g. 1, 1, 0, 1 if the value of the MPS is 1). The solid rectangles represent the A intervals, each of which is divided into the M and L sub-intervals. The MPS and LPS symbols are assumed to have fixed probability estimates of 3/4 and 1/4 respectively; hence L and M are assumed to be initialized to [0,1/4) and [1/4,1) instead of [0,1/2) and [1/2,1). (In an actual coder, the probability estimates usually start at 1/2 and are frequently updated.) At the end of the encoding sequence, C = 121/256 and A = 27/256. Therefore, C + A = 37/64; whence C' can be anywhere in [121/256,37/64). The value 1/2 lies in this interval, and is represented by a single bit, 1, with a leading binary point implied. Thus, only one bit is required in the compressed data stream for this simple example. Note that the horizontal line representing 1/2 passes through sub-intervals M, M, L and M of the four A intervals, including the initial A interval [0,1). Thus, C' = 1/2 has sufficient precision to allow correct decoding of the four symbols.

Figure A.2 shows the decoding of the four symbols with the initial value of C set equal to 1/2. Note that C is always in the sub-interval representing the decoded symbol.

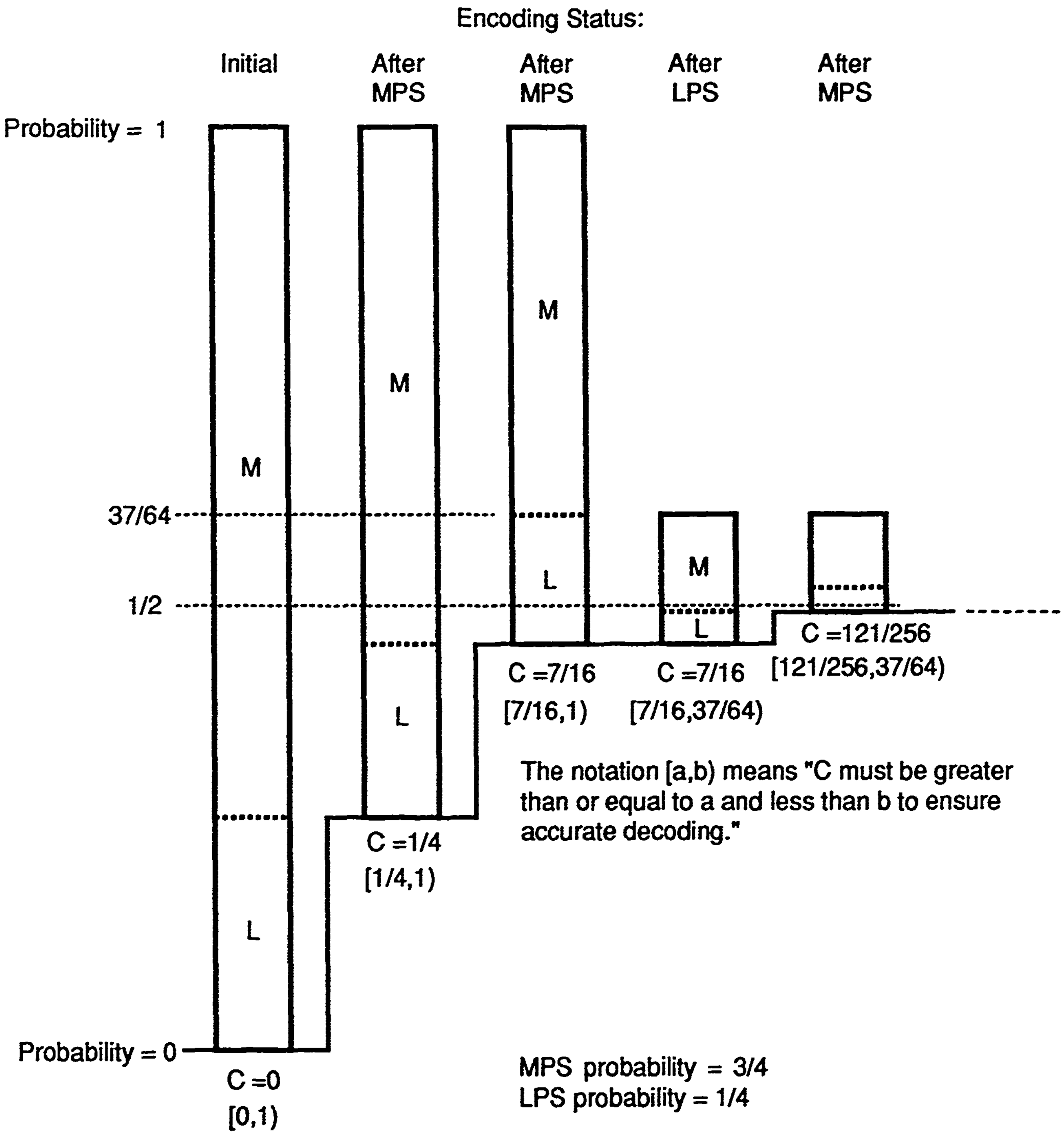
In the fictitious encoder and decoder assumed above, the values of A and its sub-intervals become smaller and smaller as the encoding/decoding process proceeds. Therefore, as the number of encoded symbols becomes large, the precision required to express these values quickly exceeds that of any practical digital processor. Moreover, the binary fraction, C, soon becomes too long to be contained in any register, or even any buffer of practical size. Leading bits of

C are therefore, in practice, output to the communications channel. This leads to the well-known carry problem: As an MPS is encoded and the LPS sub-interval is added to C, a carry might propagate through a long string of 1's into a part of C already transmitted.

Any practical encoder and decoder typically copes with these problems in the following manner:

- o The processors employ fixed-precision integer arithmetic, with implied leading binary point. The interval A and the value of C are frequently scaled up, e.g. doubled, so that precision is maintained. This scaling is usually called renormalization.
- o As C is scaled up, leading bits are periodically sent to the compressed data stream.
- o Precautions are taken to cope with the carry problem. A common method is called bit stuffing. When the encoder generates a predetermined fairly small number of successive 1's (e.g. 8), it inserts a "stuff bit" of value 0. Any carry generated later is trapped by this 0 if not to its right. The decoder, upon receiving the predetermined number of 1's, arithmetically adds the stuff bit (which can be 1 or 0, depending on whether a carry propagated into it) to the value of C already received to correct its value.[1]
- o The encoder and decoder contain identical probability estimators. The estimation methods and frequency with which the estimates are updated are based on compromises among data compression and processing speed and complexity.
- o The processing described above employs multiplication, a costly operation whether the implementation is in hardware or software. Practical systems use approximations to eliminate, or at least minimize, the multiplication burden, at the expense of less compression. Accurate decoding is assured provided that the two sub-intervals of A remain contiguous, greater than zero, and do not overlap, even if the values of the sub-intervals do not accurately represent the MPS and LPS probabilities. That is why, in the

[1] A means of producing "pure" output, without bit stuffing, is summarized in the body of this report.



The notation [a,b) means "C must be greater than or equal to a and less than b to ensure accurate decoding."

MPS probability = 3/4
LPS probability = 1/4

Figure A.1 Graphical Representation of Smple Binary Arithmetic Encoding Example

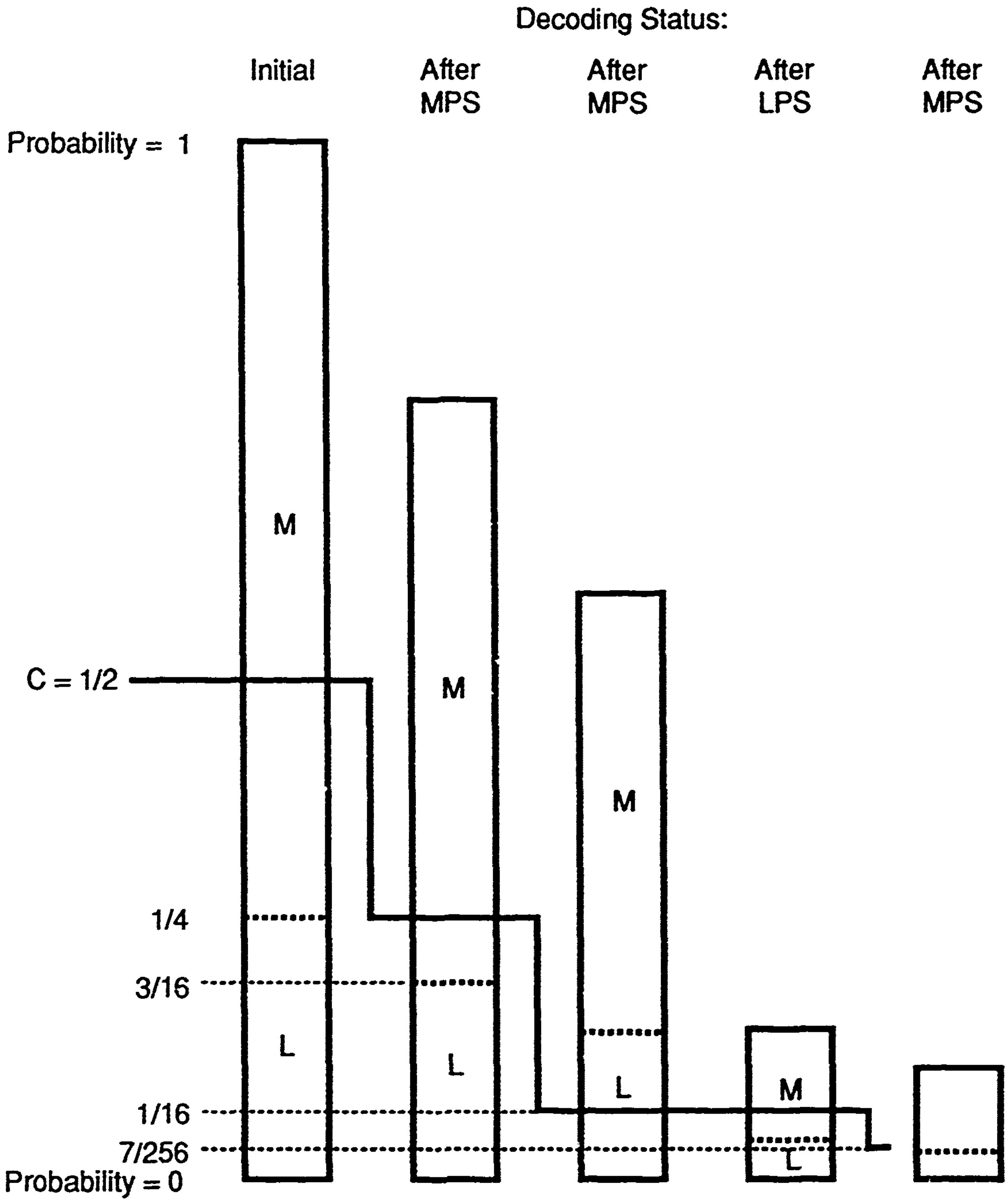


Figure A.2 Graphical Representation of Smple Binary Arithmetic Decoding Example

basic steps shown above, sub-interval M is set equal to A - L, thus ensurirg that M + L = A, even if M and L are only approximately the optimal values for best compression.

LIST OF TABLES

APPENDIX B

TEST IMAGE DISPLAYS

LIST OF FIGURES

FIGURE NO. DESCRIPTION

Kodak

September 1, 1988

Ms. Jane Doe
999 Parkside Avenue
Buffalo, NY 14214

Dear Ms. Doe:

This is to confirm our meeting of September 9, 1988. I am enclosing an Itinerary and directions for reaching the Elmgrove Plant.

We look forward to meeting you next week. If you have any questions, please feel to contact me at (555) 526-8769.

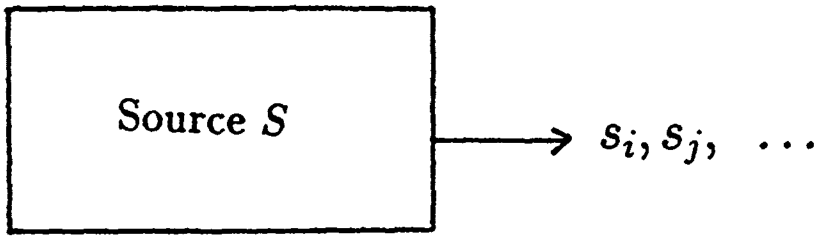
Sincerely,

Frank

Frank Weiner
Supervisor
Image Electronics Center

FXW:pat
Enclosures (2)

Information Theory Concepts



urce alphabet of size n with probabilities $p(s_1), p(s_2), \ \ldots, p(s_n)$.

he information (in bits) provided by the occurrence of urce symbol s_i is given by

$$I(s_i) = \log_2 \frac{1}{p(s_i)} \quad \text{bits}$$

he average amount of information obtained per symbol om the source is called the entropy $H(S)$ of the source:

$$H(S) = \sum_{i=0}^{n} p(s_i) \log_2 \frac{1}{p(s_i)}$$

xample:

$$S = \{A, B, C, D\}$$

$$P(A) = \frac{1}{2}, \ P(B) = \frac{1}{4}, \ P(C) = \frac{1}{8}, \ P(D) = \frac{1}{8}$$

$$H(S) = 1\frac{3}{4} \quad \text{bits}$$

銀行の「300万円MMC」登場まで1ヵ月

資産集中狙い獲得合戦し烈

[illegible]リットにわずかながら手が届く。この新型預金への都市銀行の命名は「スーパーMMC」。銀行にとっては、これまで複数の銀行、口座に小口に分散していた顧客の金融資産を、まとめて受け入れるチャンスだけに、獲得合戦はここへきてし烈さを増している。あとひと月足らず。「出遅れれば顧客が離れる」と、各行ともそれまで預金をつなぎとめるための「つなぎ商品」も開発、思い思いのネーミングで売り込みに力を入れている。

「つなぎ商品」も様々

「準備口座」や「金投資」など

都銀の「つなぎ商品」の中でいちばん多いのは、定期預金の預け替えと積立預金を組み合わせたタイプ。三菱、太陽神戸、協和、埼玉、北海道拓殖が扱っている。仕組みは、まずスーパーMMCの「準備口座」をつくり、預金者に満期がきた定期預金を、そのつど預け入れてもらう。同時に、別に毎月一定額を積み立て、合わせて「スーパーMMC」を利用できる三百万円になるのを目指す。あらかじめ満期日を設定し、予定通り三百万円以上になれば、「スーパーMMC」に振り替える。難点は、過程の金利が期日指定定期（一年物で税引き前三・三九%）のため、あまり高くない点だ。

これに対して、金投資や外貨預金など比較的利回りの高い商品を「スーパーMMC」に結びつけるところもある。

金投資口座では、三井銀行が「ゴールデンブリッジ」、住友銀行は「スーパーMMC準備コース」と命名し、販売を強化中。三井銀行では六月末満期、十月末満期の二種類を用意、利回りは年三・七—三・九八%とする。住友銀行では一—五カ月の五コースを用意、利回りは年三・三五四—四・二〇九%。どちらも「小口MMCを目指して資産形成を」というのがキャッチフレーズ。

一方、大和銀行ではスイスフラン通知預金に「シグマ」と命名、五月から取り扱いを始めた。利回りは八日現在で三カ月ものが年四・三〇五四%。九月三十日までの臨時商品。

また、第一勧業銀行では、定期預金と抵当証券を組み合わせた「ベスト」に新型を用意、預け入れが通常の一年間より

目玉をつくらず、いろいを広げているところもある和銀行は手持ちの定期預金して、振り替え日と金額をすれば六月五日以降いつ「スーパーMMC」に振りる「予約サービス」のほか預金、外貨預金、金投資を使った方法も扱う。富士の準備預金口座「トゥモロは、満期日の指定無しで、

都市銀行の主な小口MMC向け準備商品			
タイプ	銀行名	商品名	内容
準備口座型	三菱	スーパーMMC準備積立預金	期日指定定期預金と積立預金のセット
	太陽神戸	スーパー準備プラン	
	協和	スーパーMMCリレープラン	
	埼玉	ハイリターン準備プラン	
	北海道拓殖	マイウェイ	
金投資口座型	住友	スーパーMMC準備コース	1—5カ月物の5種類
	三井	ゴールデンブリッジ	6月満期と10月満期の2種
外貨預金型	大和	スイスフラン通知預金シグマ	金利は毎日変動
セット商品型	第一勧業	ベスト特別版	抵当証券と定期預金
予約型	三和	スーパーMMC予約サービス	6月5日以後スーパーMMC購入
準備口座型満期日なし	富士	トゥモロー（6月5日から取り扱い）	満期日なし。300万円で自動振り替え

[illegible]CD（譲渡性預金）の発行金利の変動に応じて、毎週預入金利の見直しを行う金融商品で、一九八五年三月に導入された。最低預入額は五千万円でスタートしたが、その後三千万円、二千万円と引き下げられ、八七年十月からは一千万円。今年六月五日からは三百万して販売される。期間は六カ月、一年の二種類。金利は六カ月物CDマイナス一・二五%、一年物CDマイナス〇・七五%になる。十月には三カ月物、二年物、三年物もスタートする。

今後、最低預入額を三百万円からさらに引き下げることが検討される予定。

対中鋼材輸出

価格据え置き 鉄鋼大手、

新日本製鉄など鉄鋼大手六社が進めていた中国への今年下期の鋼材輸出商談は八日、輸出価格を据え置く形で決着した。

国際市況の上昇を理由にげを主張していた日本側と、内のインフレ抑制のために下げを求めていた中国側がしたわけだが、上期分の中

日米通信摩擦で郵政次官 改めて「譲歩せぬ」

市場分野別（MOSS）協議の日米合意違反だとして、米政府が対日制裁を決めた電気通信問題について、奥山郵政事務次官は八日の記者会見のなかで「すでに米側に示した回答以上に譲る考えはない」と、新たな譲歩はしない方針を改めて強調した。また、三塚通産相が同日、この問題について二十七日までに米政府に「中間回答」をする考えを示したことに対し、同次官は「今のところそういう予定はない。郵政省が責任をもって対処する」と述べた。

奥山次官らは同日、竹下首相に会い、同省の方針の説明や現状報告を行ったが、首相から特別の指示はなかったという。また、片岡郵政相は同日までに、三塚通産相から訪米の報告を電話で聞いたほか、自民党の[illegible]幹事長とは電話で、[illegible]長には直接会い「日本はMOSS合意に違反していない」と説明し、同省の姿勢に対する[illegible]を求めた。

さらに同省は、[illegible]相の[illegible]に同行した[illegible]幹部から[illegible]に説明を聞く予定。これらの[illegible]を通して同省は、「[illegible]分野は郵政省の所管の日本[illegible]OSS合意に違反していないことを政府の統一見解にしえで、対応策を検討する方

民間設備投資

今年度は10.6%増

2年連続2ケタ台

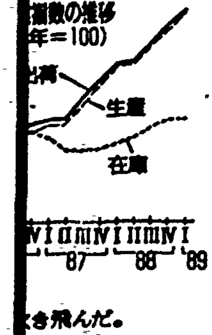
産構審の部会調査

拡大が新たな設備投資を生む「設備投資循環」②技術革新の進展③消費の多様化による高付加価値化促進④労働時間短縮への対応⑤各種規制緩和に向けた競争力確保——の五点を挙げて

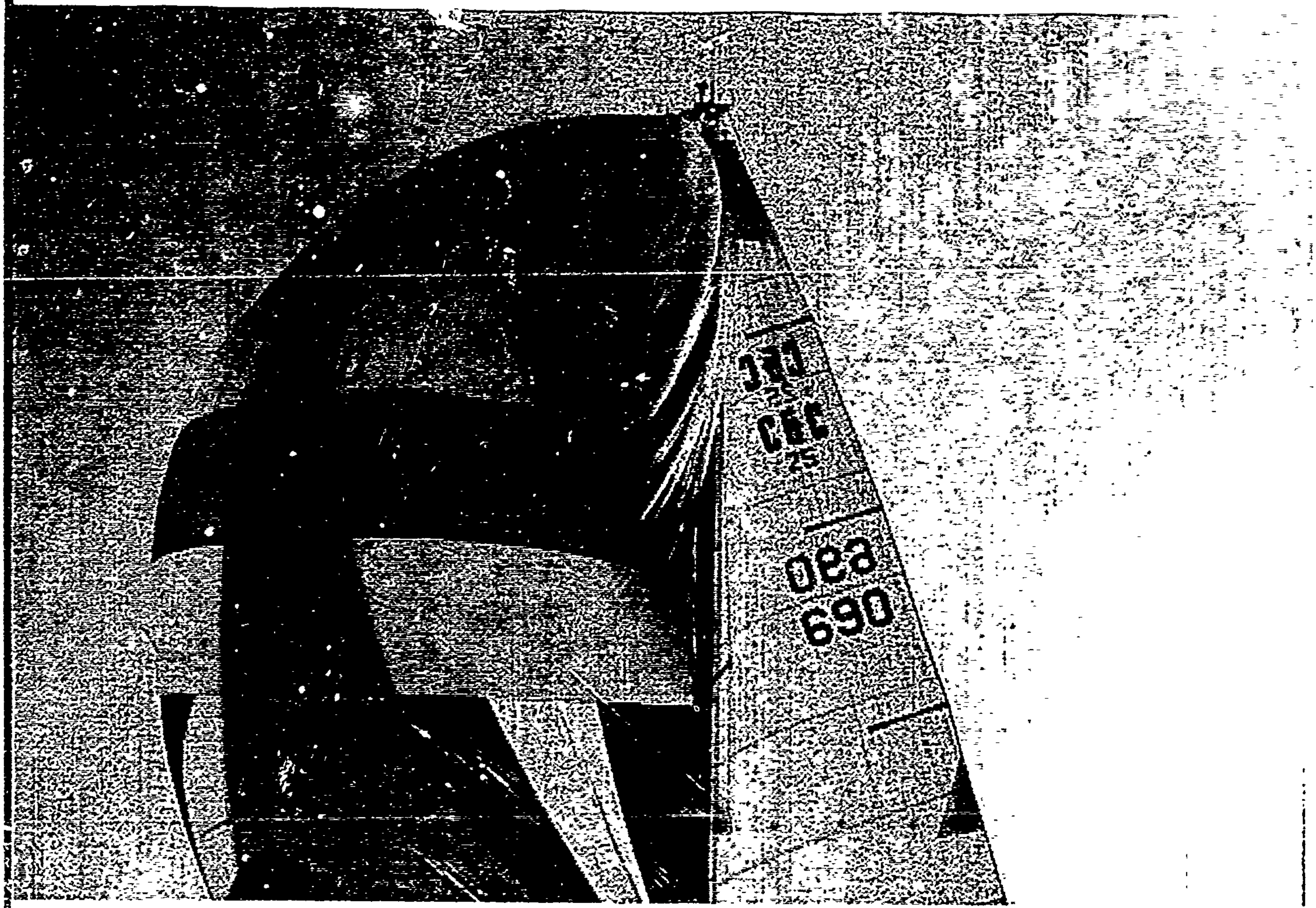
、・は速報。通貨）
CD、平均残高

3月の景気動向をす統計数字がおおたそろったが、単でみると爆発的といえる景気上昇をしている。消費税入前のかけこみ需のあった百貨店販額は史上最高を記、乗用車販売もとも絶好調。国内い。1、2月は昭去の影響で個人消りが見えたが、す

指数の推移
年＝100）



き飛んだ。
や、鉱工業生産指
は前月比4.2%の
67年5月以来の記


CBC
75
690

March 16, 1989

弓野さん

いろいろと ありましたが
最終的には Ektachrome 135シリーズ（アマ用）を
取り寄せて. Yuminoさんへ 送ることになりました.
これは. 届いていないとはいえ 送り返されたことは 事実であるので
Director's Officeに 承知してもらいましたので 安心して下さい.

この件で 電話をしていたら 石田さんが やって来て 説明してくれて
私には やっと 135シリーズ というのは アマチュア用のことである, etc
少しわかってきたところです.

他のことで 精神的に お疲れのところ おさわがせしましたが
これで一件落着です

ただし 入荷（EKJ RADに）するのが 週明け頃ですので
Pouchにて Yuminoさんに 届くのは 10日後（今日から）くらいに
なりそうです. 御了承下さい.

Blumen wiegen sich im Reigen,
legen an ihr schönstes Kleid,
wollen sich der Sonne zeigen
nach der langen Winterszeit.

Und der Frühling küßt die Blüten,
sie erschauern bis ins Glück,
wie Geheimnis, das sie hüten,
kehrt nur ihr Traum zurück.

by Gertrud Arnold

"Er geht ja in der Wäsche noch ein, Tante", erwiderte Jossie
nft und wandte sich um, denn die Tür zum Lokal wurde
öffnet, und der erste Gast dieses Tages trat ein. Sie war
nen Augenblick so verblüfft, daß sie nicht einmal guten
ig sagen konnte. Dafür sagte Taubert es, und sie erkannte
fort die Stimme, die sie am Telefon gehört hatte. Eines
ar allerdings komisch: Sie hatte ein gutes Gedächtnis für
esichter, aber seins war noch nie hier im Sonnebachhöf-
en aufgetaucht. Aber anderswo, dachte sie - anderswo.
och es fiel ihr nicht so schnell ein, wo es gewesen sein
onnte.

Sie kommen um Ihre Mappe?" sagte sie.

Erraten! Und Sie sind Fräulein Jossie Sonnebach und waren
o liebenswürdig, mich anzurufen."

Ja! - Ihr Niersteiner ist kalt gestellt, Herr Doktor!"

Er lachte hellauf und Matnilde Sonnebach, die sich in der
wischenzeit ebenfalls umgestellt hatte, weil nämlich die-
r Doktor durchaus kein alter Herr war, glaubte bemerken
u müssen, daß man im Sonnebachhöfchen sehr wohl wisse,
it welcher Temperatur man den Weißwein zu kredenzen
abe. "Zu kalt ist nämlich auch bei Weißwein von Übel",
rklärte sie.

"Wie. überhaupt bei allem", erwiderte er mit einem ver-
teckten Lächeln. "Am schlimmsten aber ist es, wenn man
n mit Sprudel mischt."

etzt wußte Jossie mit einemmal, woher sie ihn kannte. Er
atte, als sie damals mit Geheimrat Heltmann in der Wein-
ube in München saß, an einem der Nebentische saure Leber
egessen, nein, gebackene, korrigierte sie ihr Erinnern. Es gab
einen Zweifel, daß er es gewesen war. Es gab auch keinen
weifel, dao er hier im Sonnebachhöfchen zum ersten Mal
nkehrte. Wie das mit der Mappe zusammenhing, mußte
ch erst herauskristallisieren. Auf jeden Fall stand sein

Berlin feiert in beiden I
dung. Im Schöneberger
heitsglocke lauten, wen
beth II. von Großbritan
Reagan und der franzö
nach Berlin kommen. B
scheidplatz an der Berli
damm.

n. Es konnte auch sein, d
den und nicht mehr zurü
llen würde schon stimmen.
überließ den Gast der Nichte
hm den bestellten Niersteine
n selber an den Fenstertisc
"so erfrischendes Lüftchen
dünnen Strohgläser daneben
genheiten benutzt wurden.
anstoßen?" fragte Dr. Taub
in schmeckt er mir nicht
etwas gefährlich, wenn ma
n muß.
ich zu zieren, ein zweites G
e voll und stieß mit ihm an
ut, daß ich Ihren Namen in
"Ich hätte sonst gar nich

immer praktisch", erwidert
nicht meine Sache. Aber
nan leicht sein Eigentum ver
ie. Sie könnte der Lust, ih
er noch nie hier gewesen
ste ihn entrüstet an, als er
ch so!" sagte sie ärgerlich.
r meine Mappe hierher?"
Sie strafen sich selber Lüg

sie. "Darauf trinke ich
Ihnen an, wenn Sie gestehe

te", sagte er, sein Weinglas
ämlich wirklich nicht, wie
Ich kann mir nur denken,
meiner Brüder einmal hier im Sonnebachhöfche
über den Durst getrunken hat und dann in se
seligkeit vergaß, die Mappe an sich zu nehmen
Verdienst bin ich nun der Nutznießer dieser Ver
Sie soll gesegnet sein! Prost, Fräulein Jossie!"

"Prosit!" sagte sie belustigt, stieß mit ihm an
dann die Mappe aus dem Einbauschrank heru
Sie, hier steht Ihr Name eingezeichnet - cand.med.

"Selige Zeiten!" lachte Taubert. "Das Leder ha
ausgehalten."

Jossie nickte, spielte mit dem Verschluß der
das etwas verrostete Schloß auf- und zuspringe
dann von neuem an.

Taubert nahm sie ihr behutsam aus den Hände
sie neben sich auf das Fensterbrett. "Sie werden
denken können, wieso ich am Telefon gesagt ha
Niersteiner ohne Sprudel möchte. - Ich habe
einer Weinstube in München gesessen, als Sie m
teren Herrn hereinkamen. Und der bestellte so
mit und Sie haben ihn deswegen getadelt und br

"Ich erinnere mich", sagte Jossie.

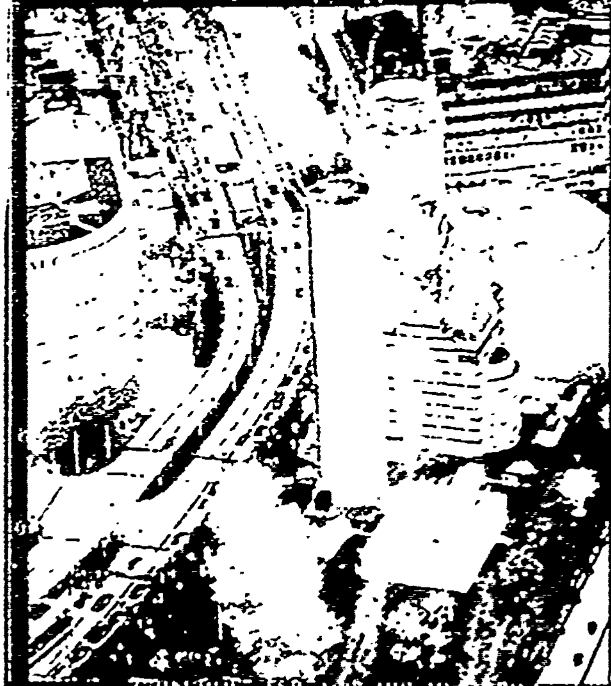
"Erinnern Sie sich auch meiner?"

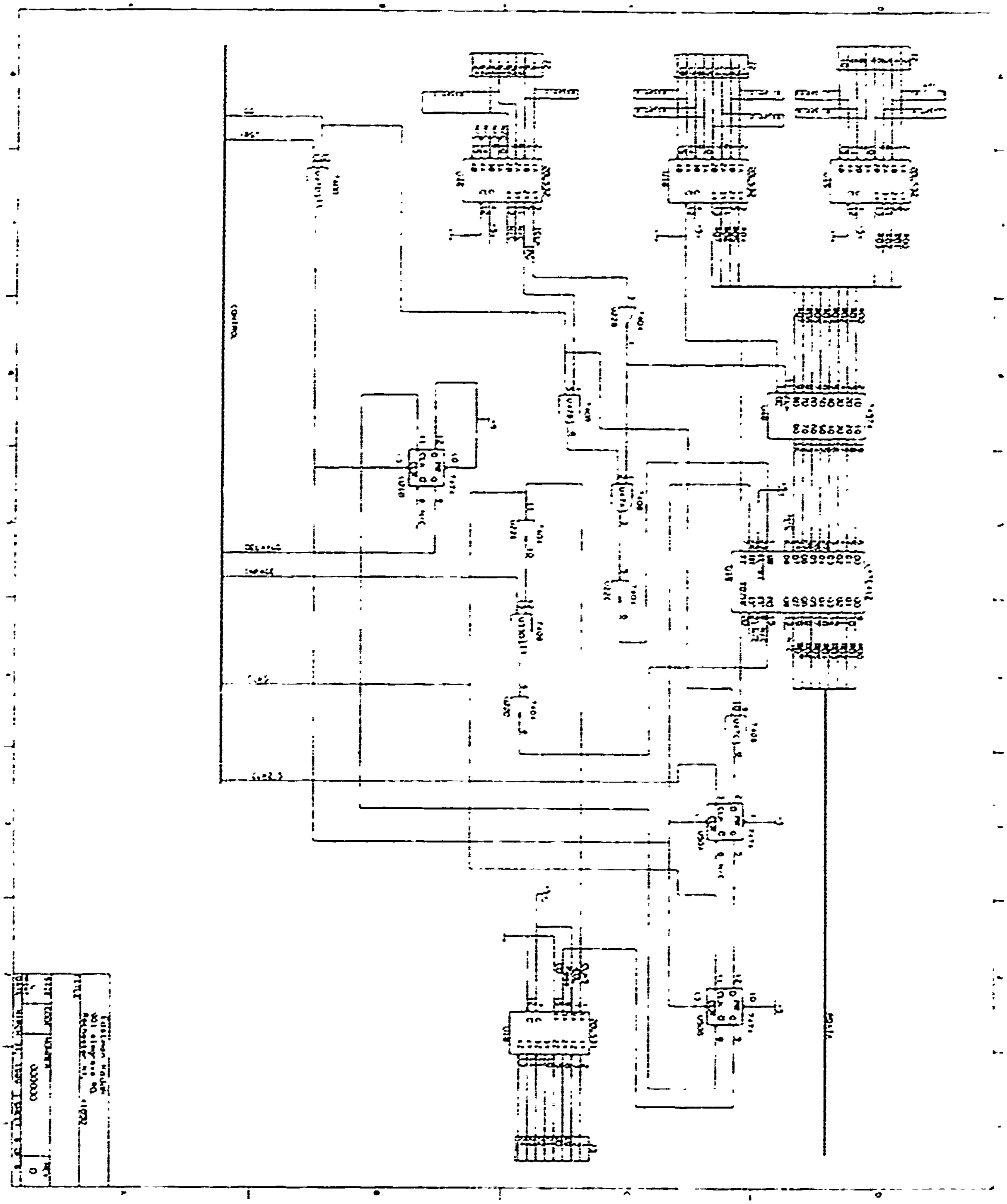
"Auch Ihrer, ja! - Sie saßen an einem der Nebe
aßen gebackene Leber

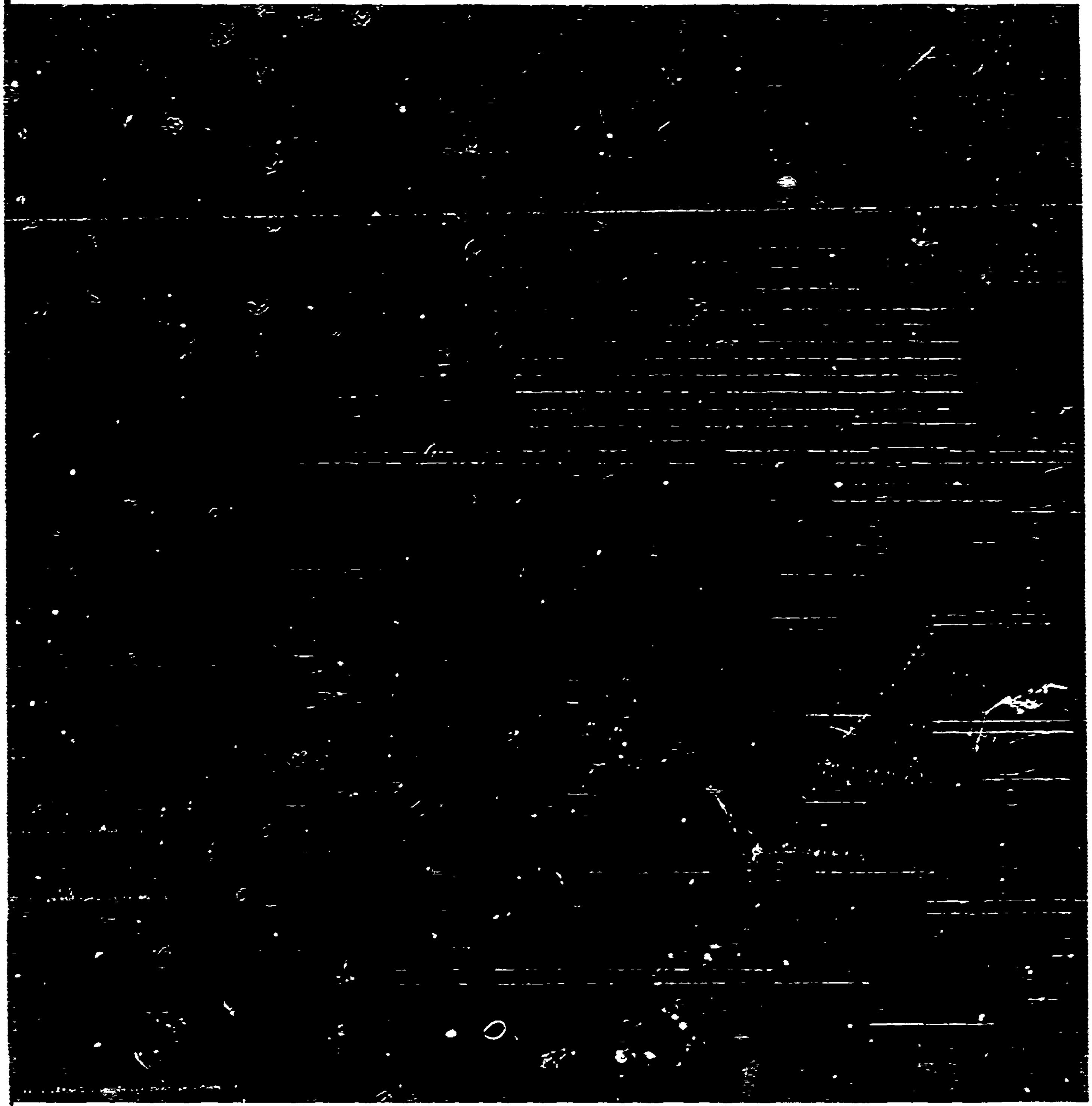
"Und bis ich auf- und mich umsah, waren Sie
Taubert. "Aber der Himmel hat mir diesen reize
mit der Mappe geschickt, und auf den trinken wi
Gott, ob ich Sie sonst so schnell gefunden hätte

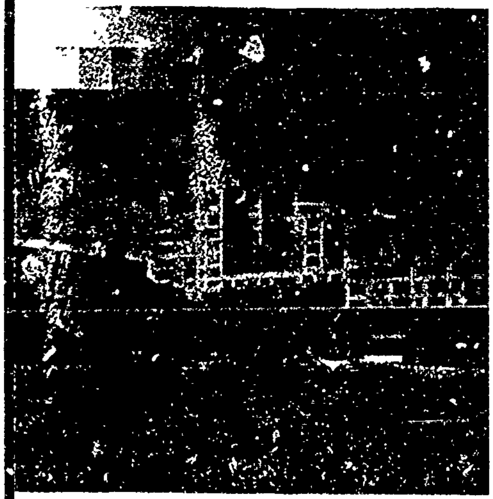
Jossie wollte fragen "haben Sie mich denn gesu
ließ es am besten sein. Er war nicht der erste
bestimmt auch nicht der letzte sein, bei dem
Flämmchen der Zuneigung aufglimmen sah. Es b
ebenso schnell wieder herunter und erlosch, ehe e
geben vermochte. Das war weder etwas Neues,

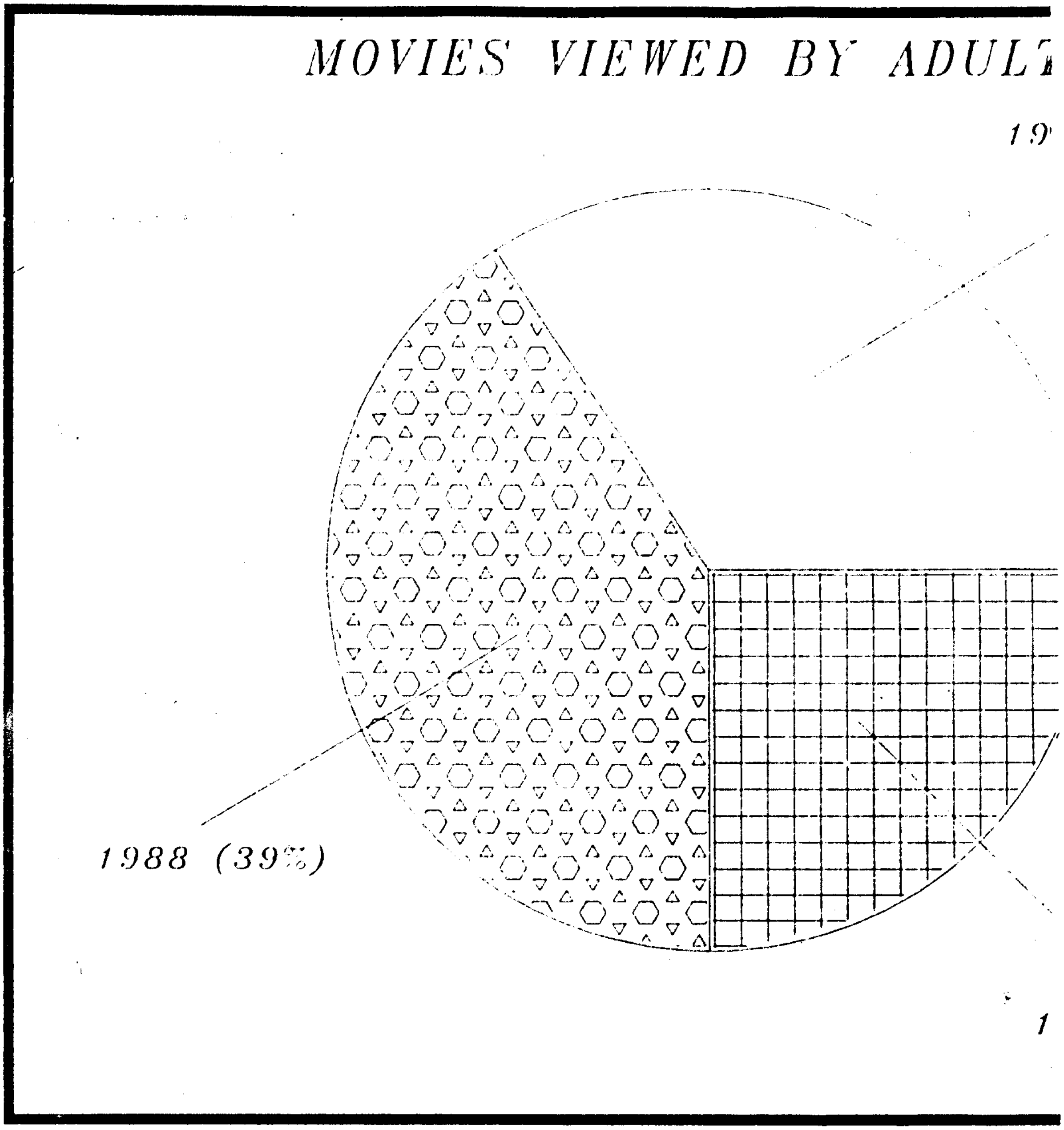
"Er geht ja in der Wäsche noch ein, Tante", erwiderte Jossie
sanft und wandte sich um, denn die Tür zum Lokal wurde
geöffnet, und der erste Gast dieses Tages trat ein. Sie war
einen Augenblick so verblüfft, daß sie nicht einmal guten
Tag sagen konnte. Dafür sagte Taubert es, und sie erkannte
sofort die Stimme, die sie am Telefon gehört hatte. Eines
war allerdings komisch: Sie hatte ein gutes Gedächtnis für
Gesichter, aber seins war noch nie hier im Sonnebachhof-
dachte sie - anderswo.
o es gewesen sein
nnebach und waren
Doktor!"
ach, die sich in der
weil nämlich die-
glaubte bemerken
en sehr wohl wisse,
wein zu kredenzen
eißwein von Übel",
er mit einem ver-
r ist es, wenn man
r sie ihn kannte. Er
mann in der Wein-
entische saure Leber
ihr Erinnern. Es gab
Es gab auch keinen
en zum ersten Mal
ammenhing, mußte
sich erst herauskristallisieren. Auf jeden Fall stand sein
Name darin gestanden hatte ihm oder hatte ihm
gehört, war abhanden oder von ihm selber verschenkt

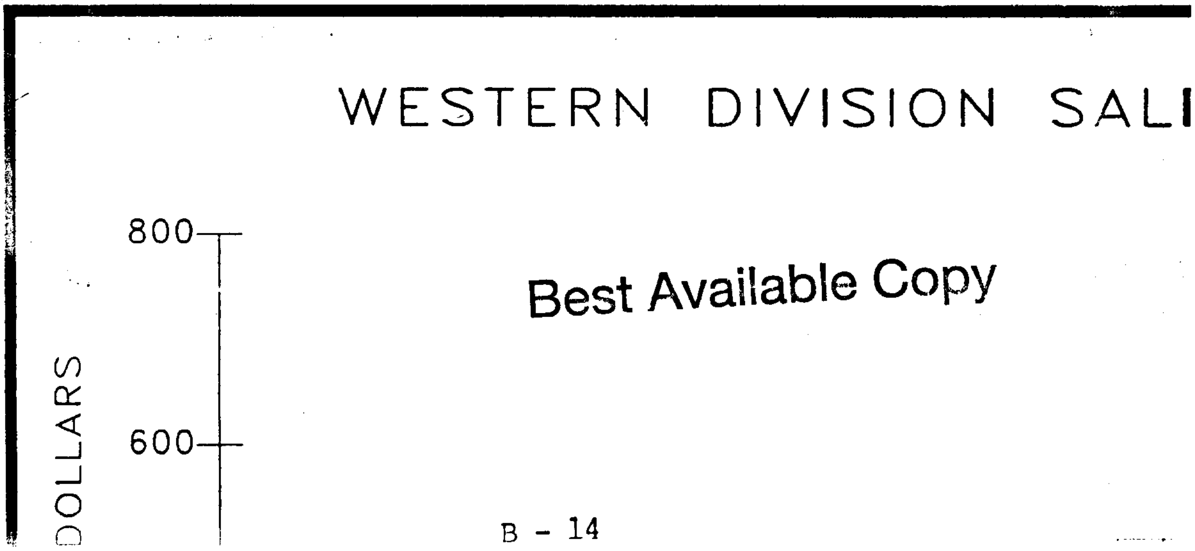









MOVIES VIEWED BY ADULT
19
1988 (39%)
1


WESTERN DIVISION SALI
800
600
DOLLARS